МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчёт о лабораторной работе №7 по дисциплине основы программной инженерии

Выполнила:
Нестеренко Тамара Антоновна,
2 курс, группа ПИЖ-б-о-20-1,
Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2021 г.

ВЫПОЛНЕНИЕ

1. Практическая часть

```
my_list = [1, 2, 3, 4, 5]
my_list = ['один', 'два', 'три', 'четыре', 'пять']
my_list = ['один', 10, 2,25, [5, 15], 'пять']

third_elem = my_list[2]
```

Рисунок 1 – Пример создания списка

```
my_list = ['один', 'два', 'три', 'четыре', 'пять']
for elem in my_list:
    print(elem)

my_list = [1, 2, 3, 4, 5]
for i in range(len(my_list)):
    my_list[i]+=5
    print(my_list)

my_list = ['один', 10, 2,25, [5, 15], 'пять']
print(len(my_list))
```

Рисунок 2 – Пример прохода (итерации) по списку

```
>>> a = [10, 20, 30, 40]
>>> for i, item in enumerate(a):
...     print(f"({i}, {item})")
...
(0, 10)
(1, 20)
(2, 30)
(3, 40)
>>>
```

Рисунок 3 – Пример прохода (итерации) по списку

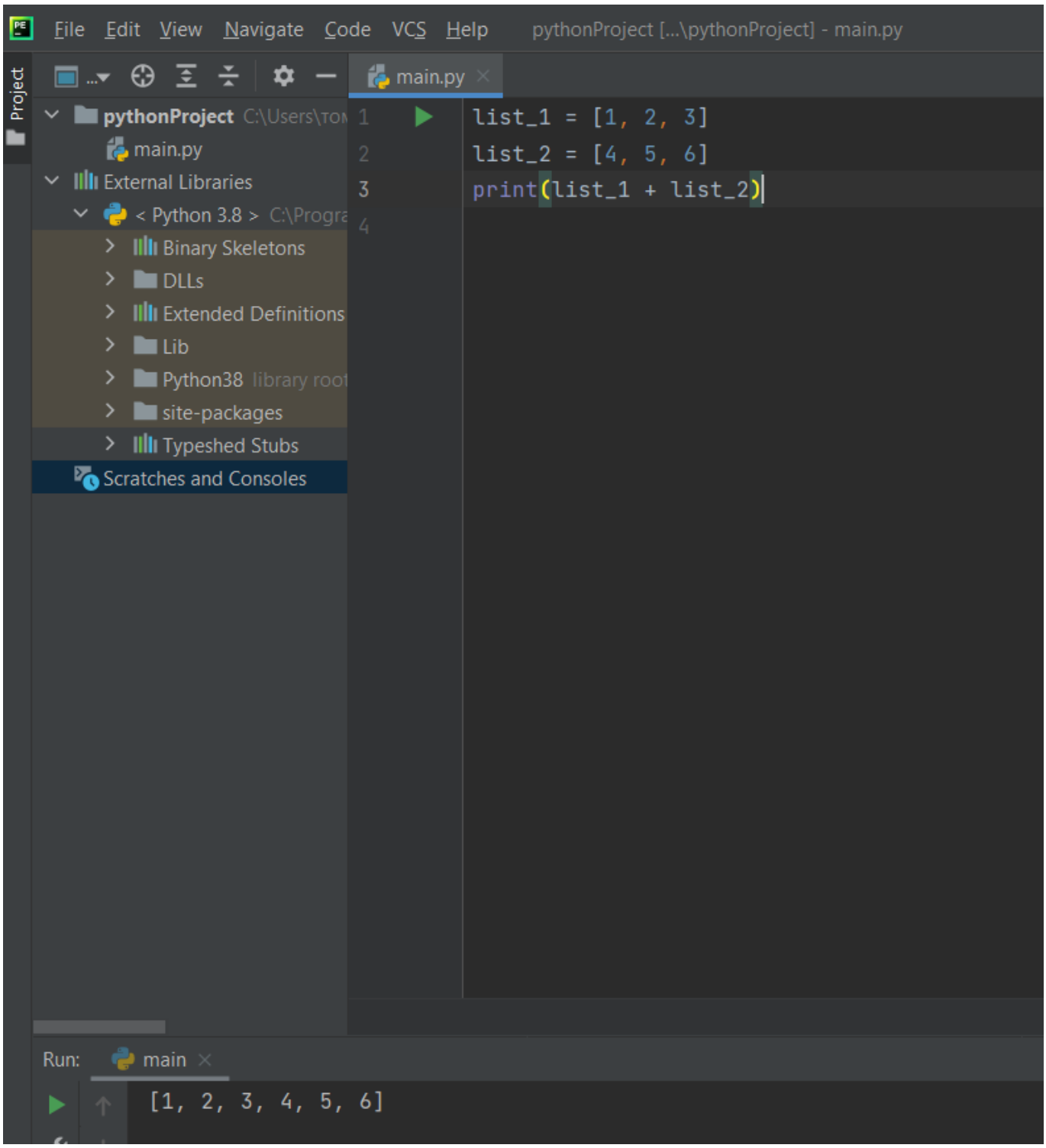

Рисунок 4 – Пример операции сложения со списками

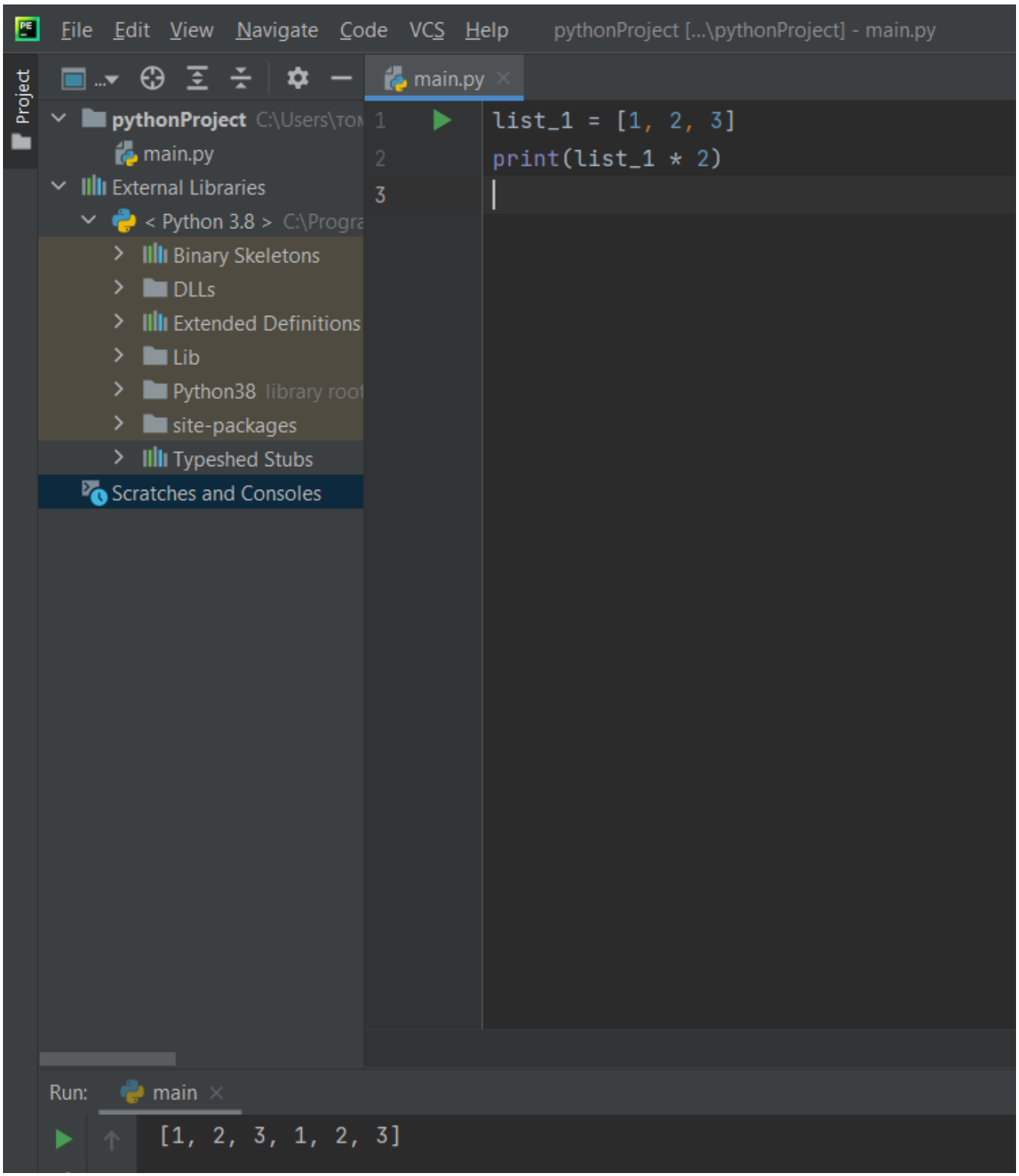

Рисунок 5 – Пример операции умножения

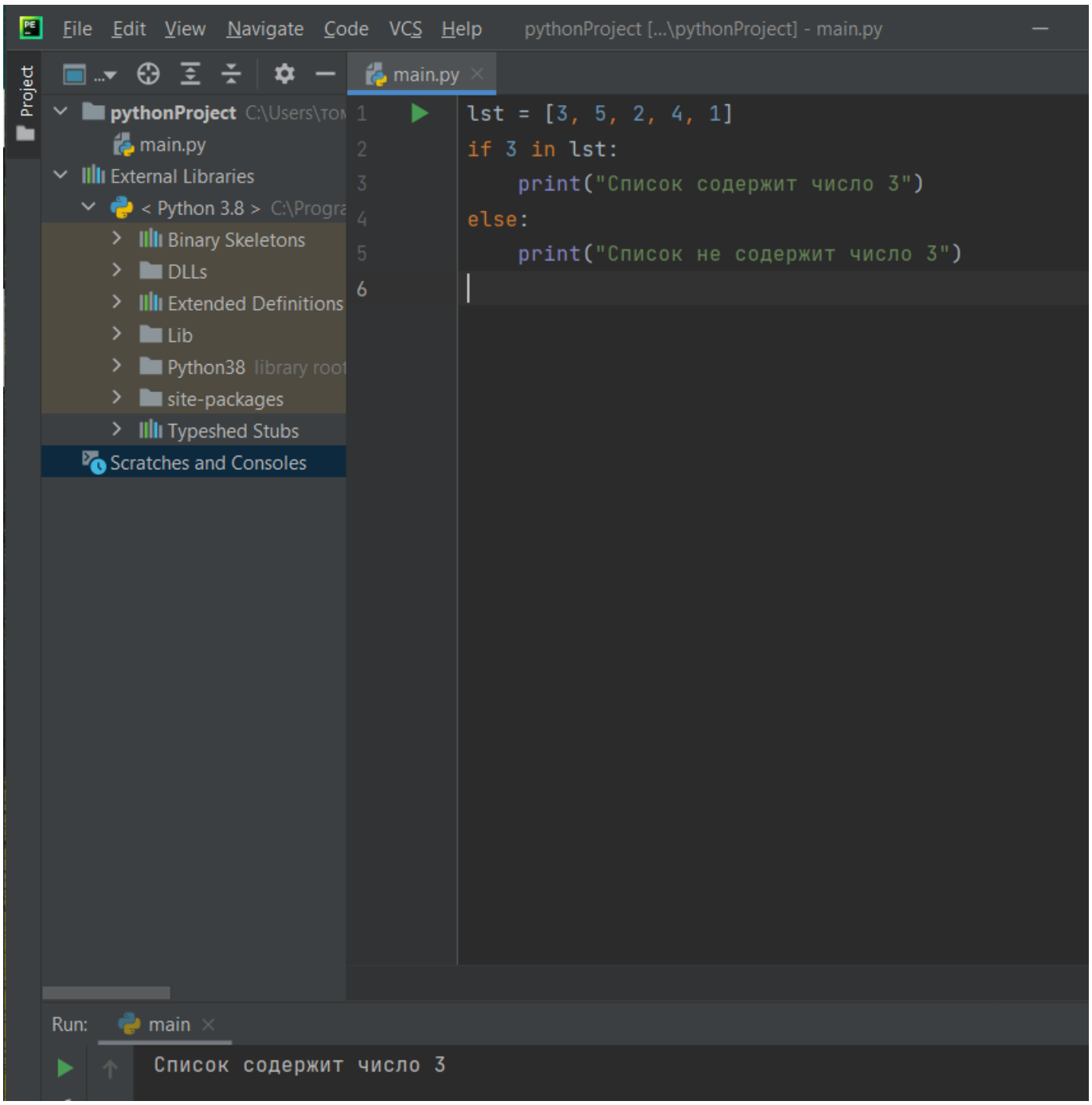

Рисунок 6 – Пример поиска элемента в списке

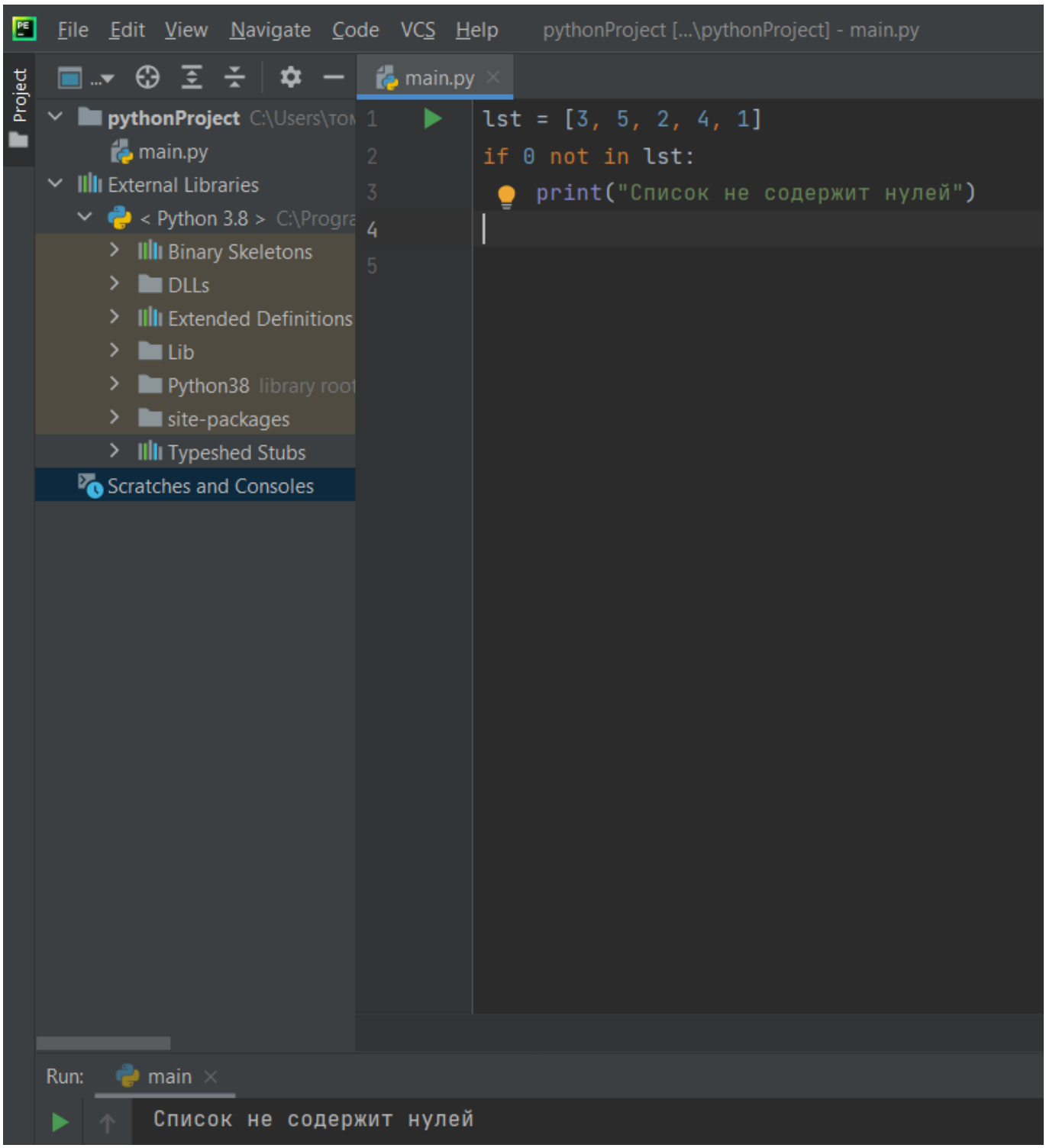

Рисунок 7 – Пример поиска элемента в списке с помощью оператора not

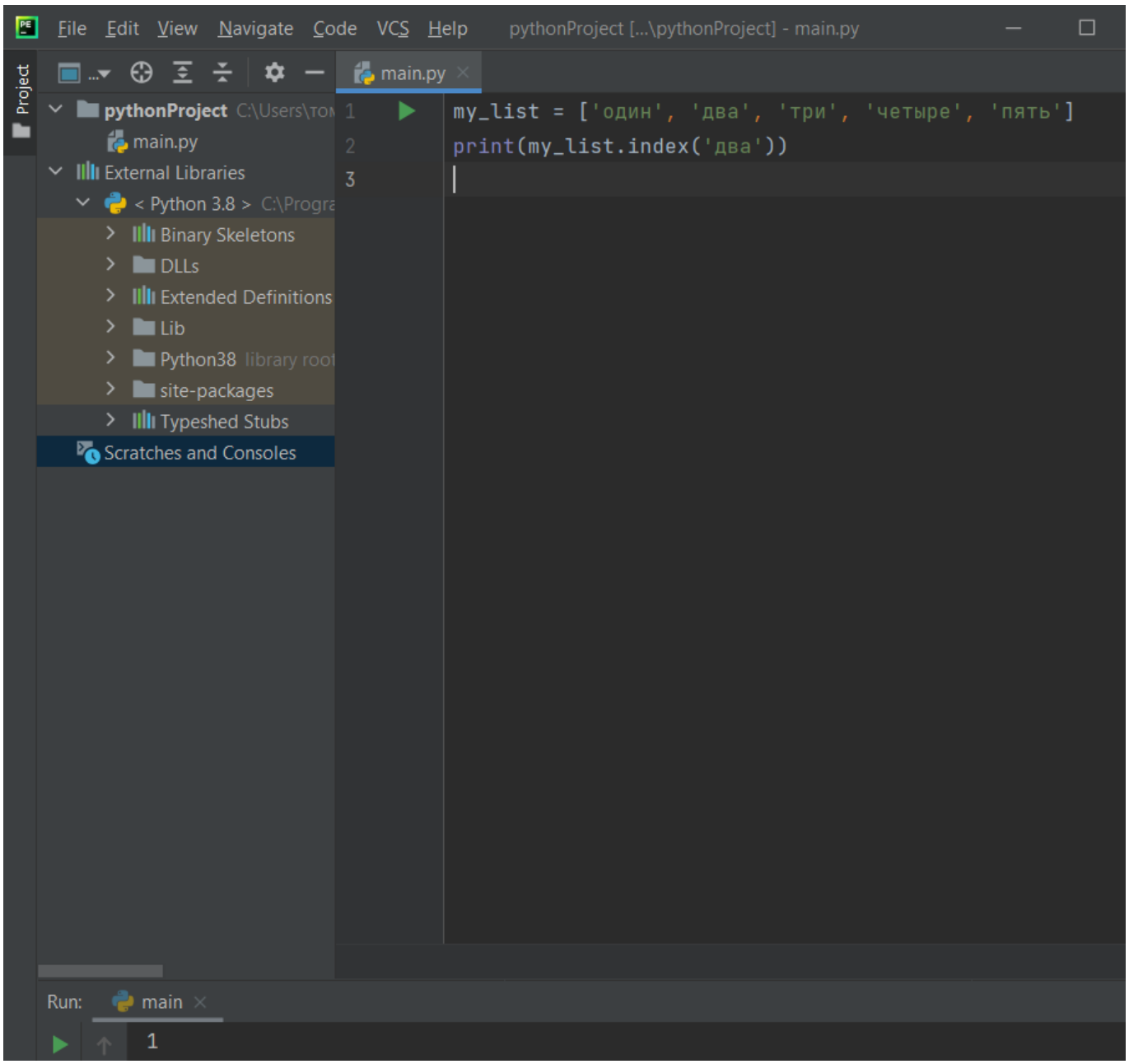

Рисунок 8 – Пример индекса элемента в списке

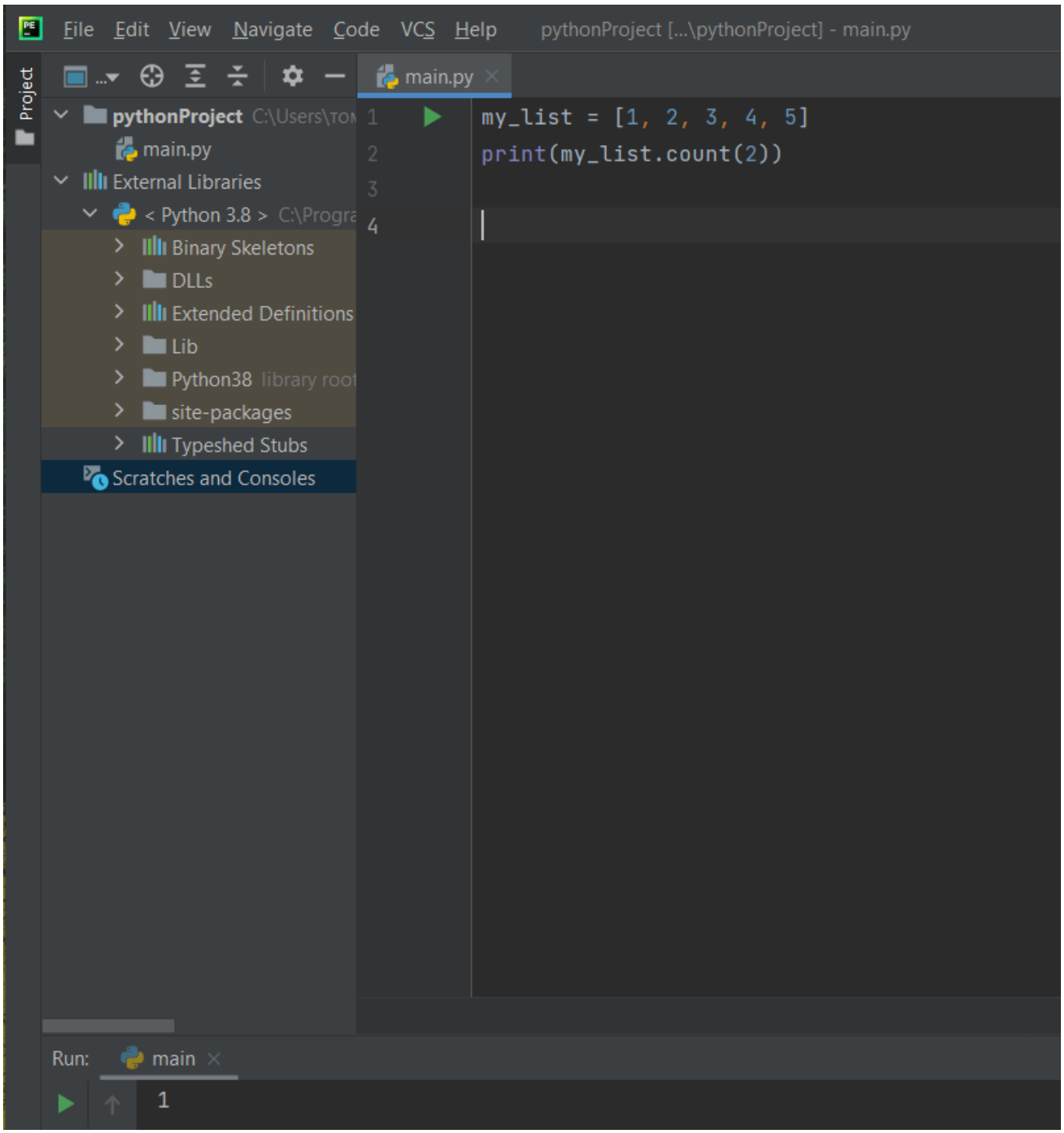

Рисунок 9 – Пример вывода числа вхождений элемента в список

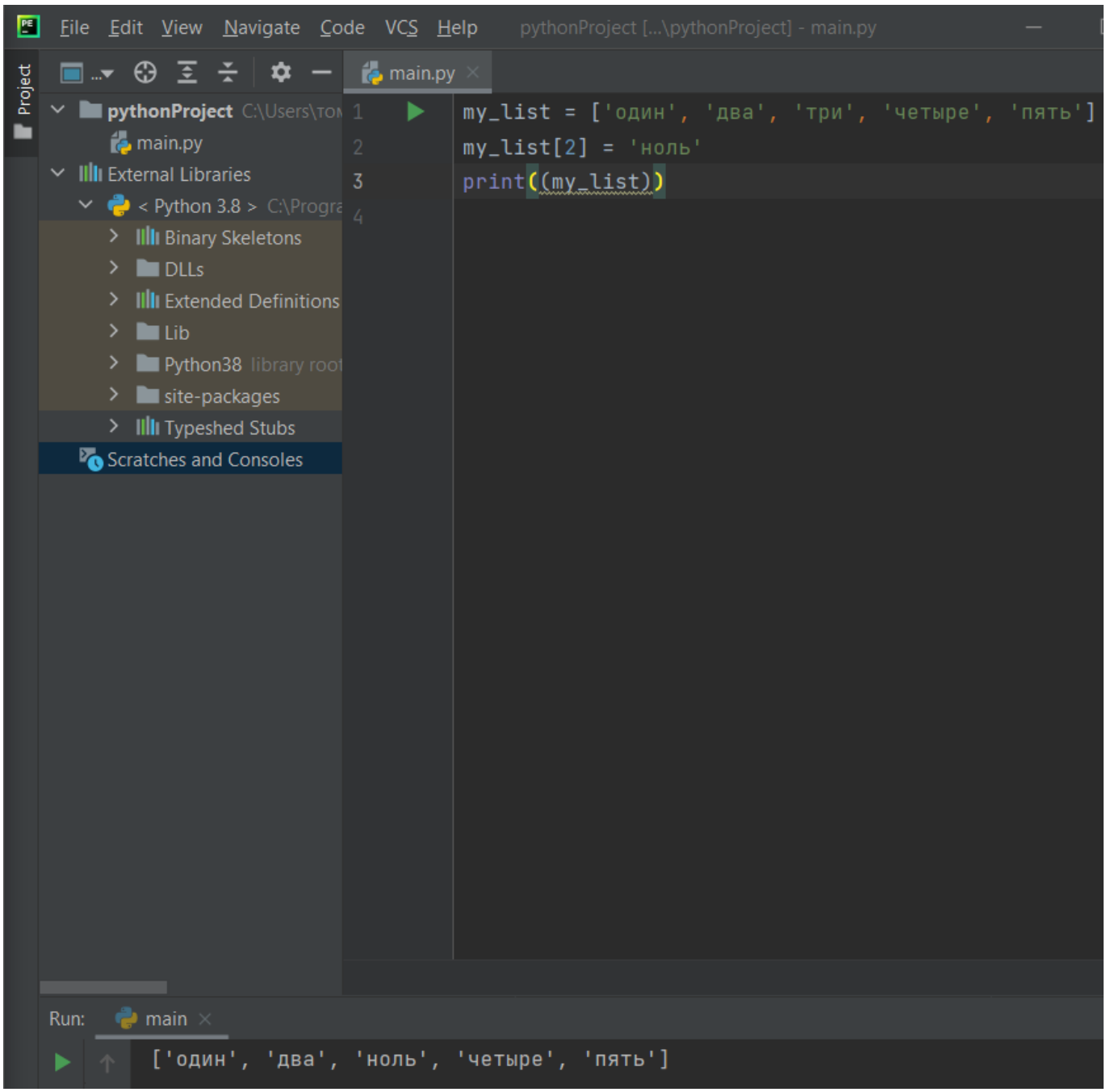

Рисунок 10 – Пример изменения списка

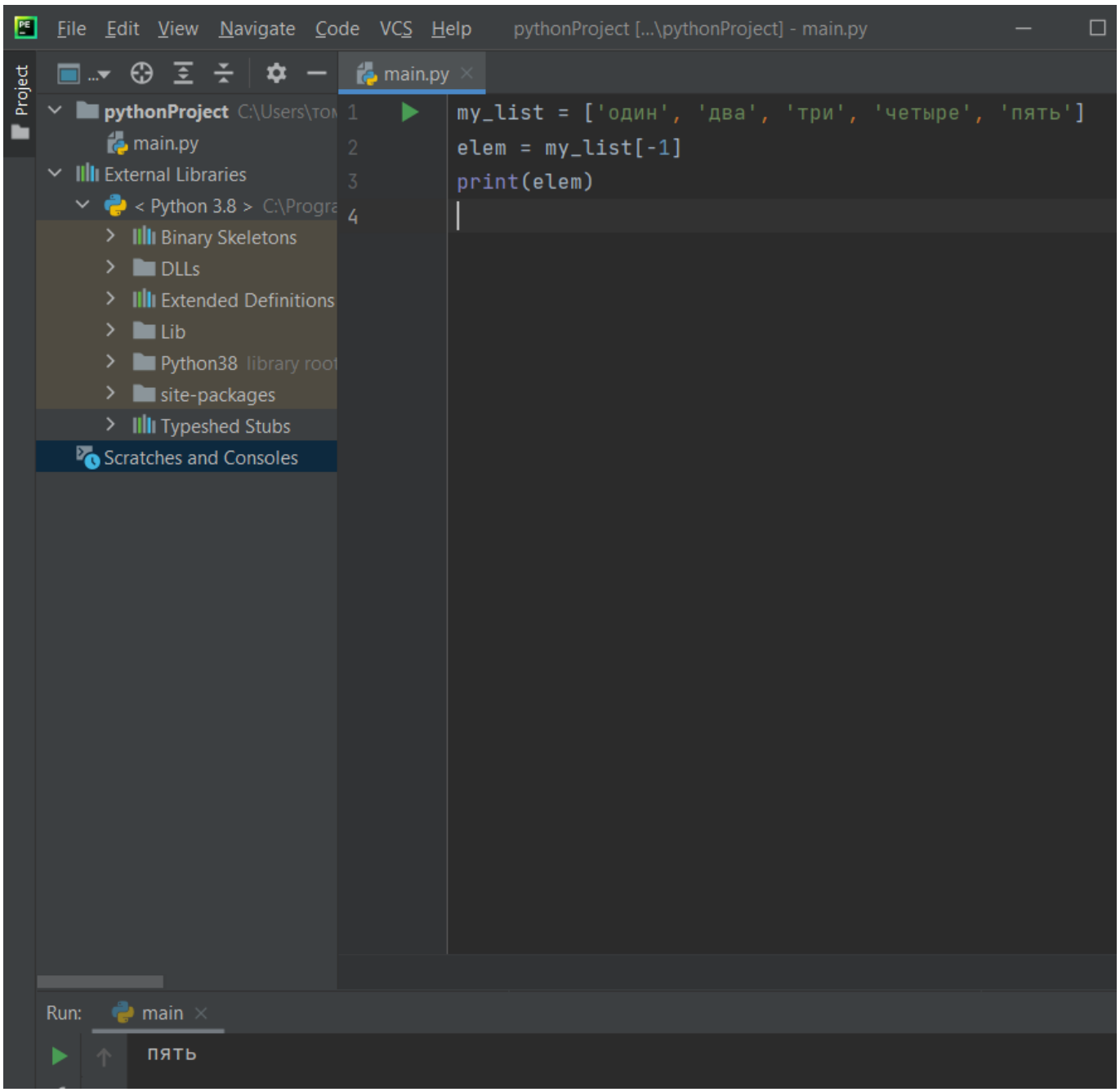

Рисунок 11 – Пример вывода элемента списка с отрицательным индексом

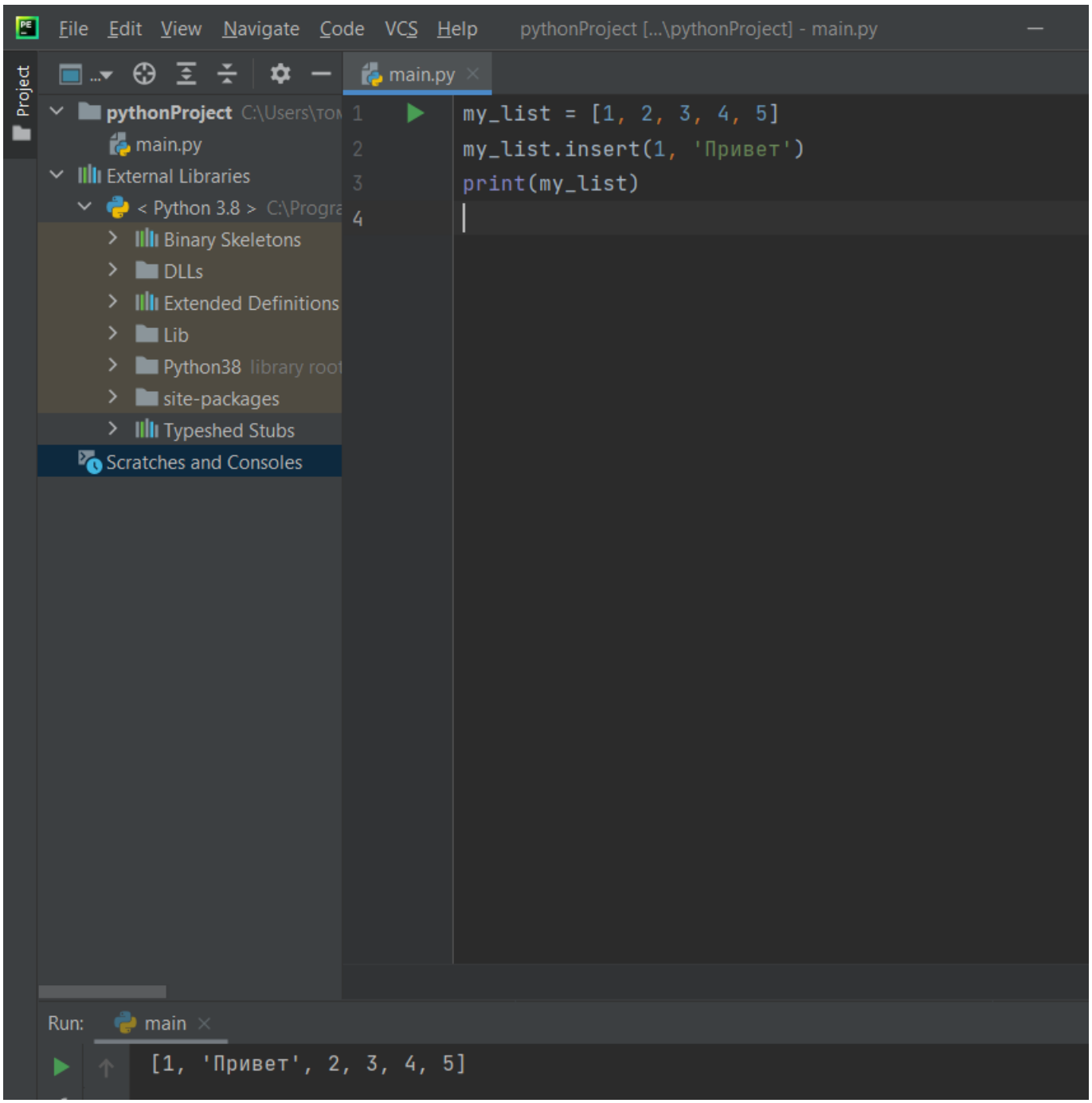

Рисунок 12 – Пример вставки элемента в список

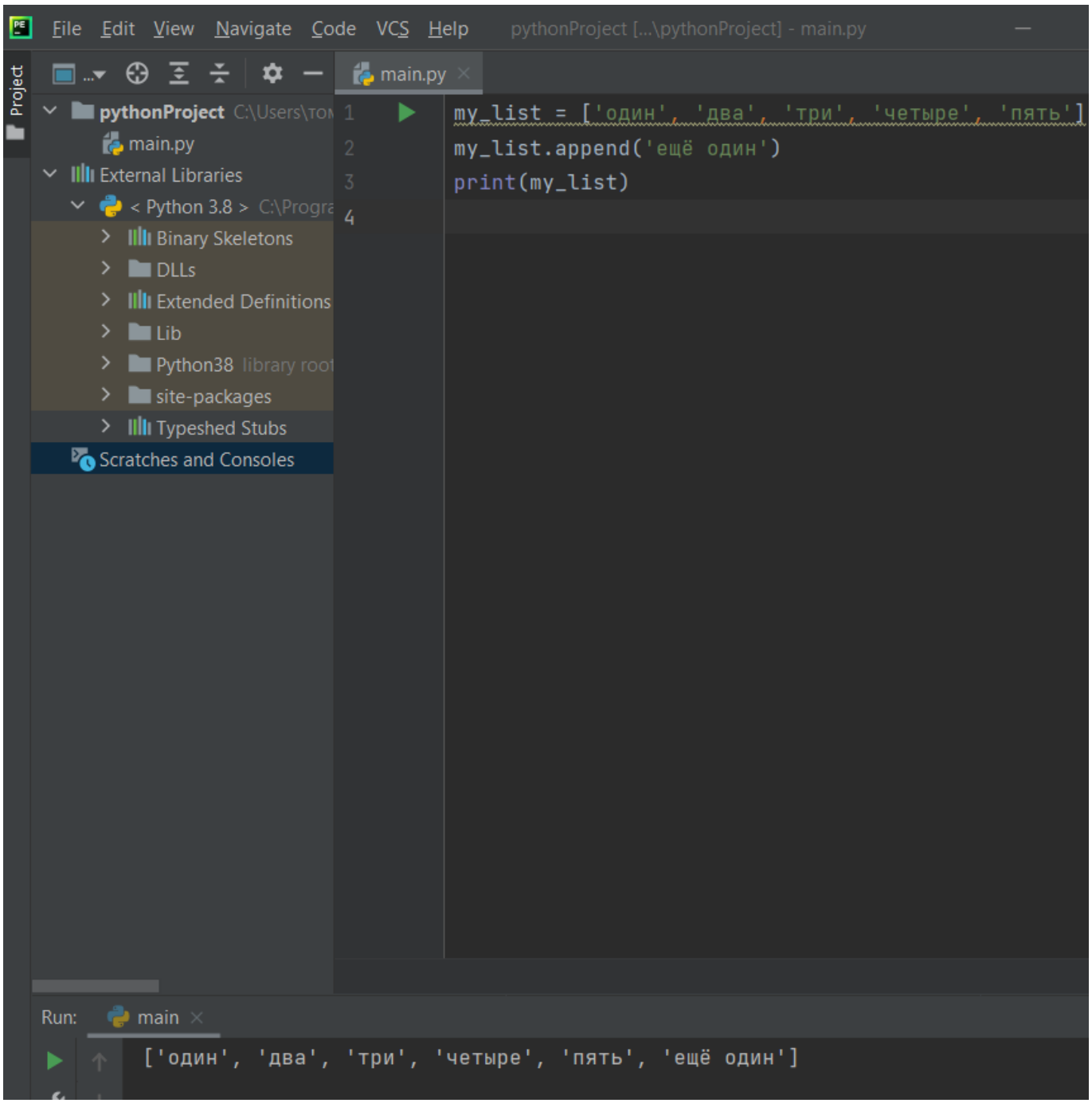

Рисунок 13 – Пример добавления элемента в список

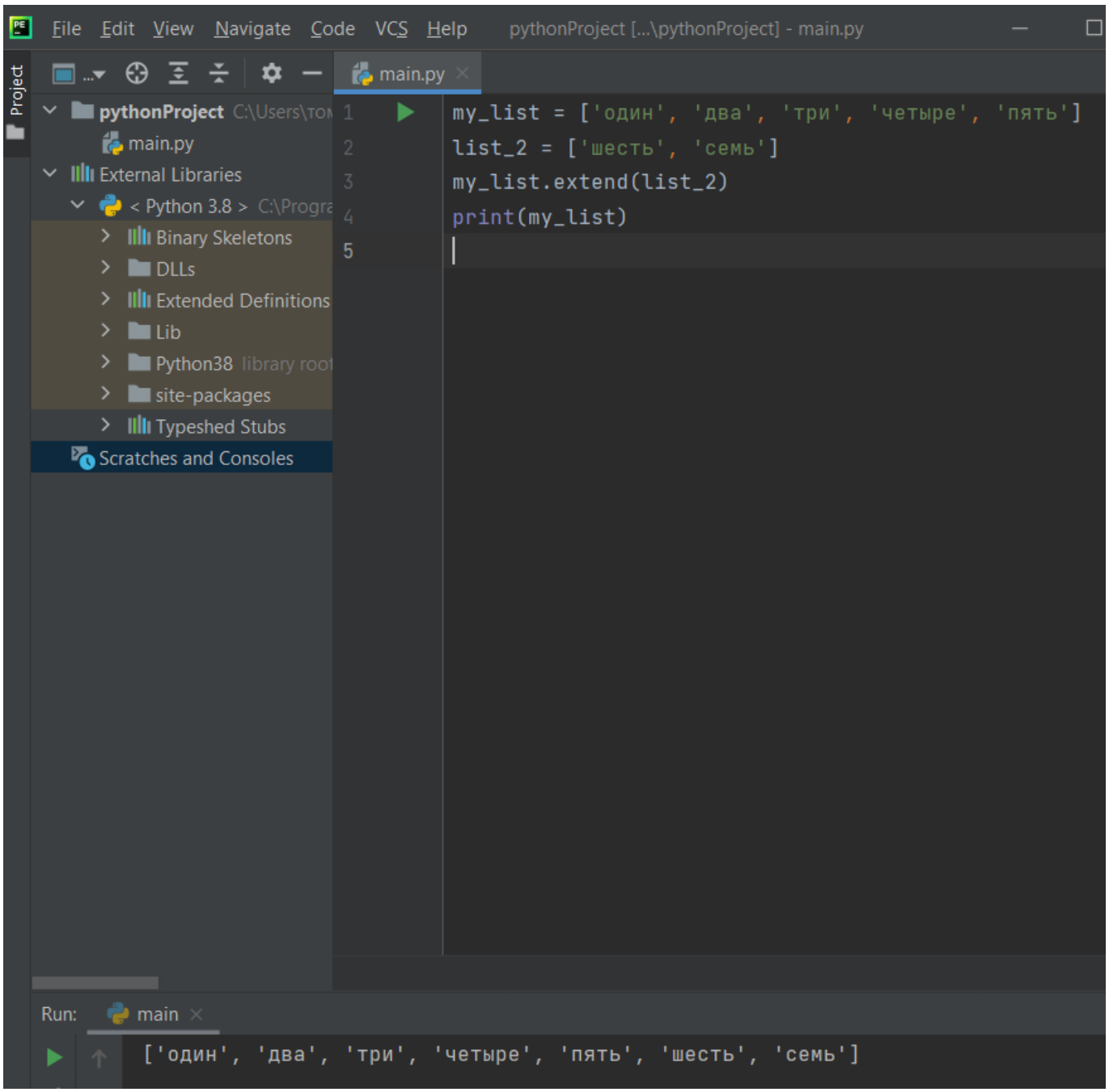

Рисунок 14 – Пример добавления элементов в список, объединяя два списка

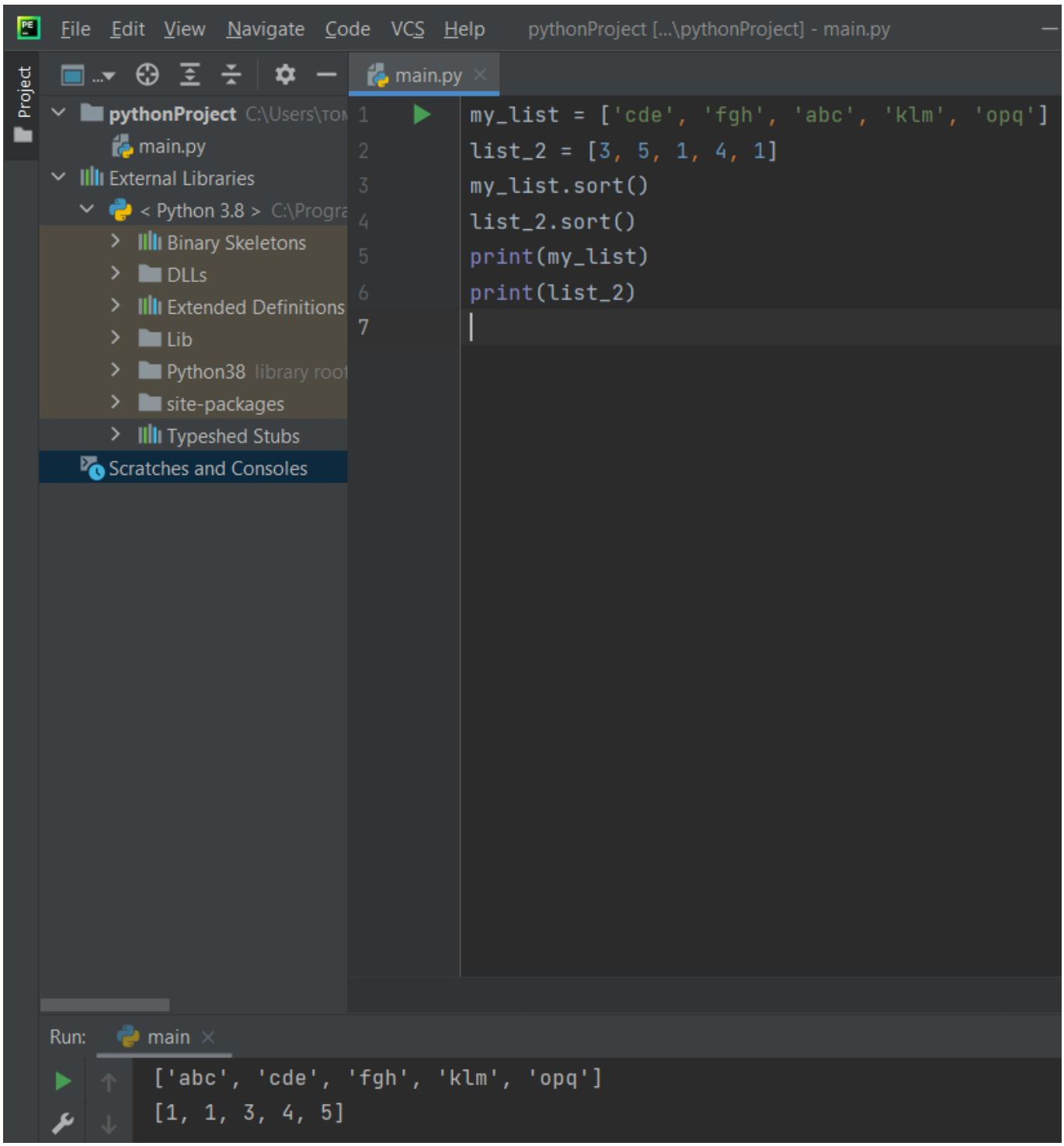

Рисунок 15 – Пример сортировки элементов

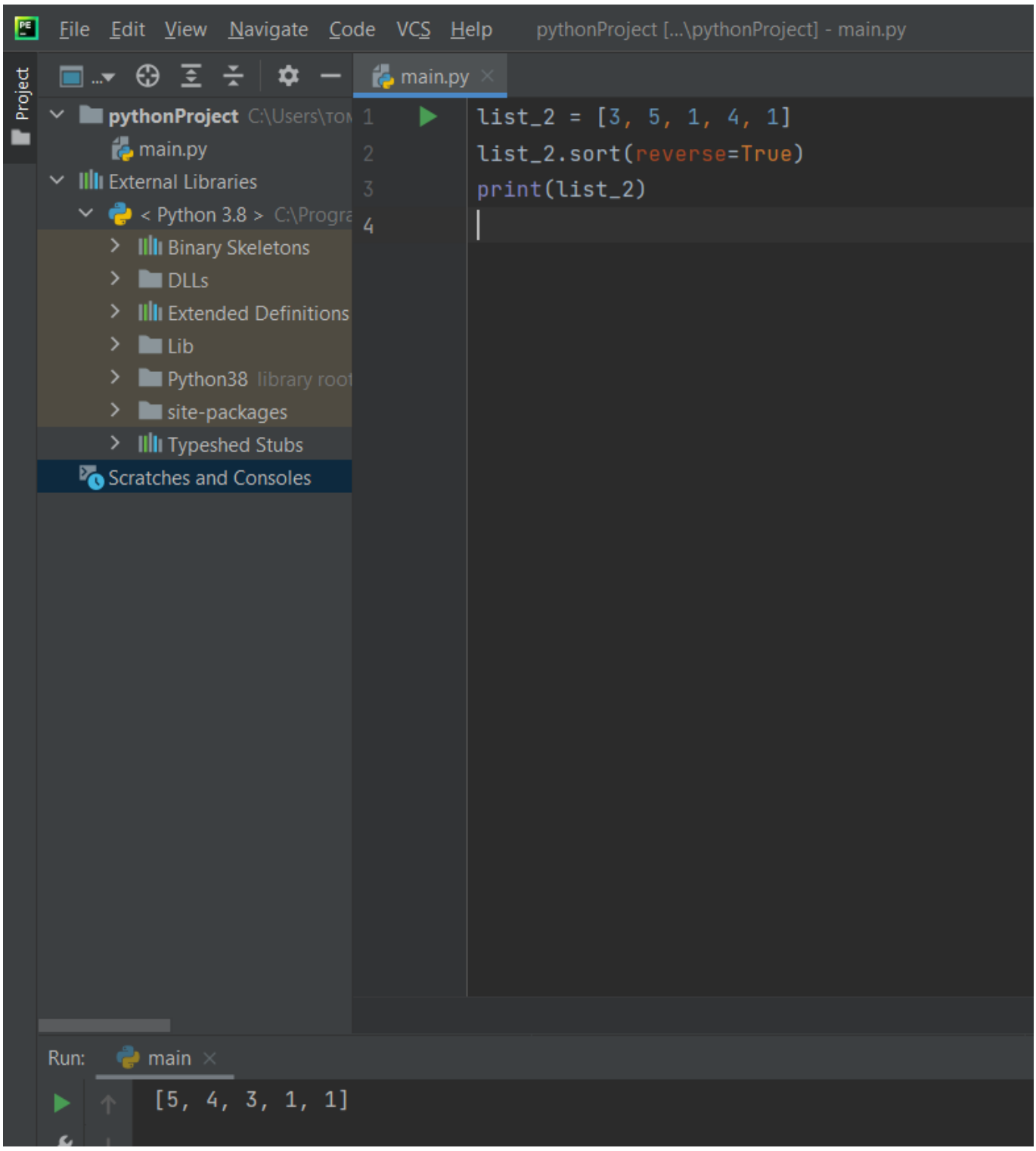

Рисунок 16 – Пример сортировки чисел по убыванию

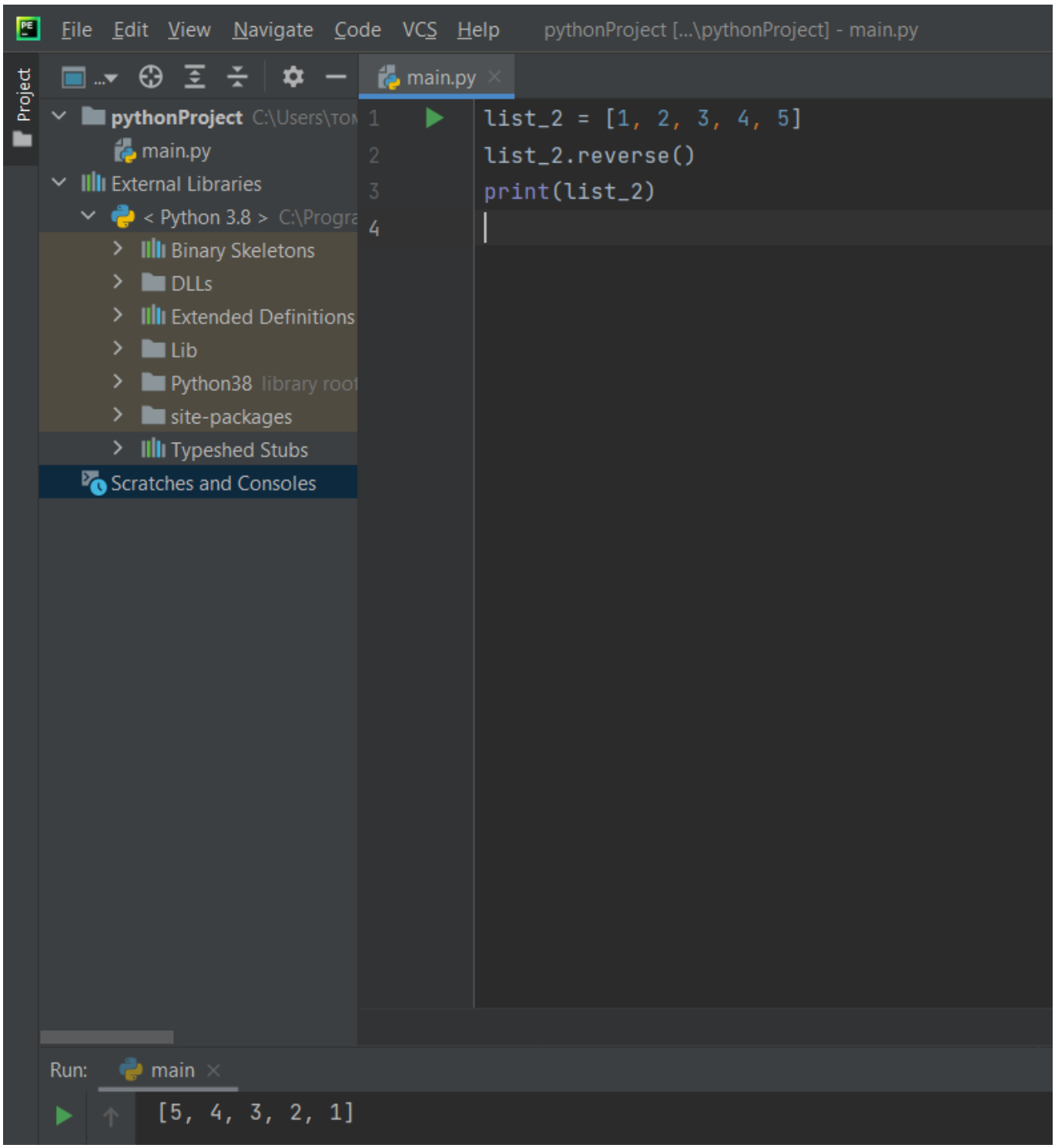

Рисунок 17 – Пример перевёрнутого списка

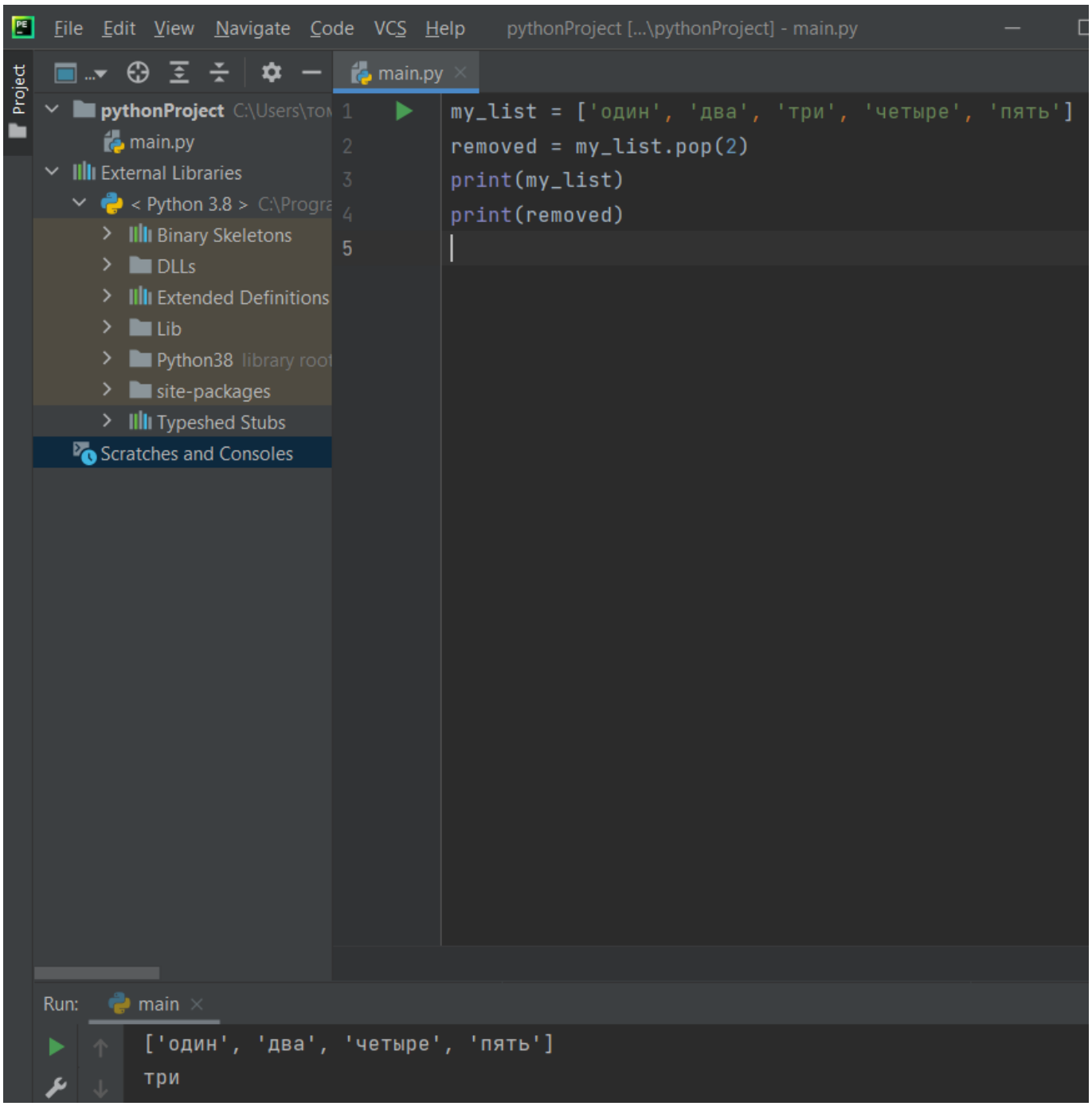

Рисунок 18 – Пример удаление элементов из списка

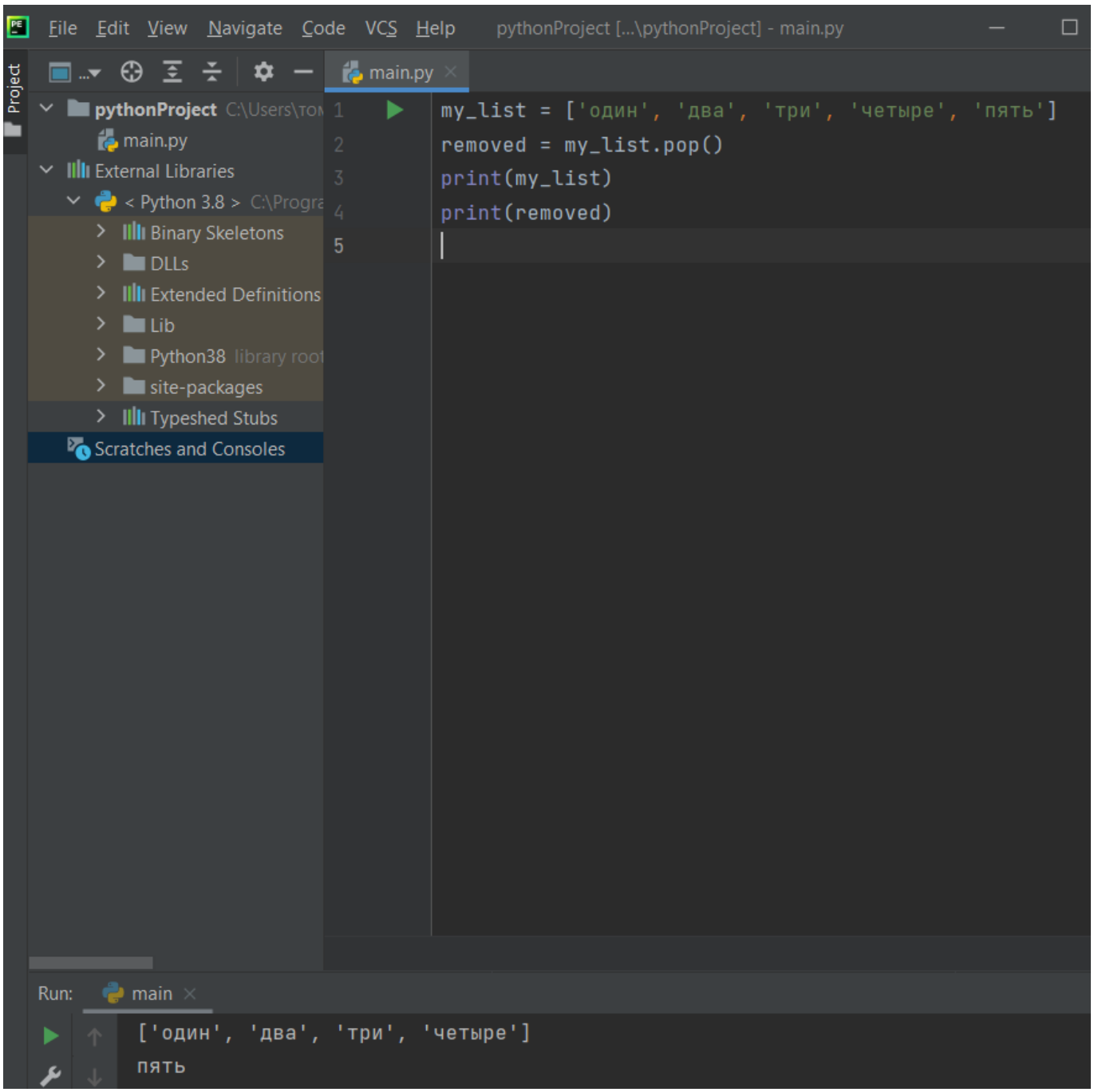

Рисунок 19 – Пример удаления элемента не указывать индекс

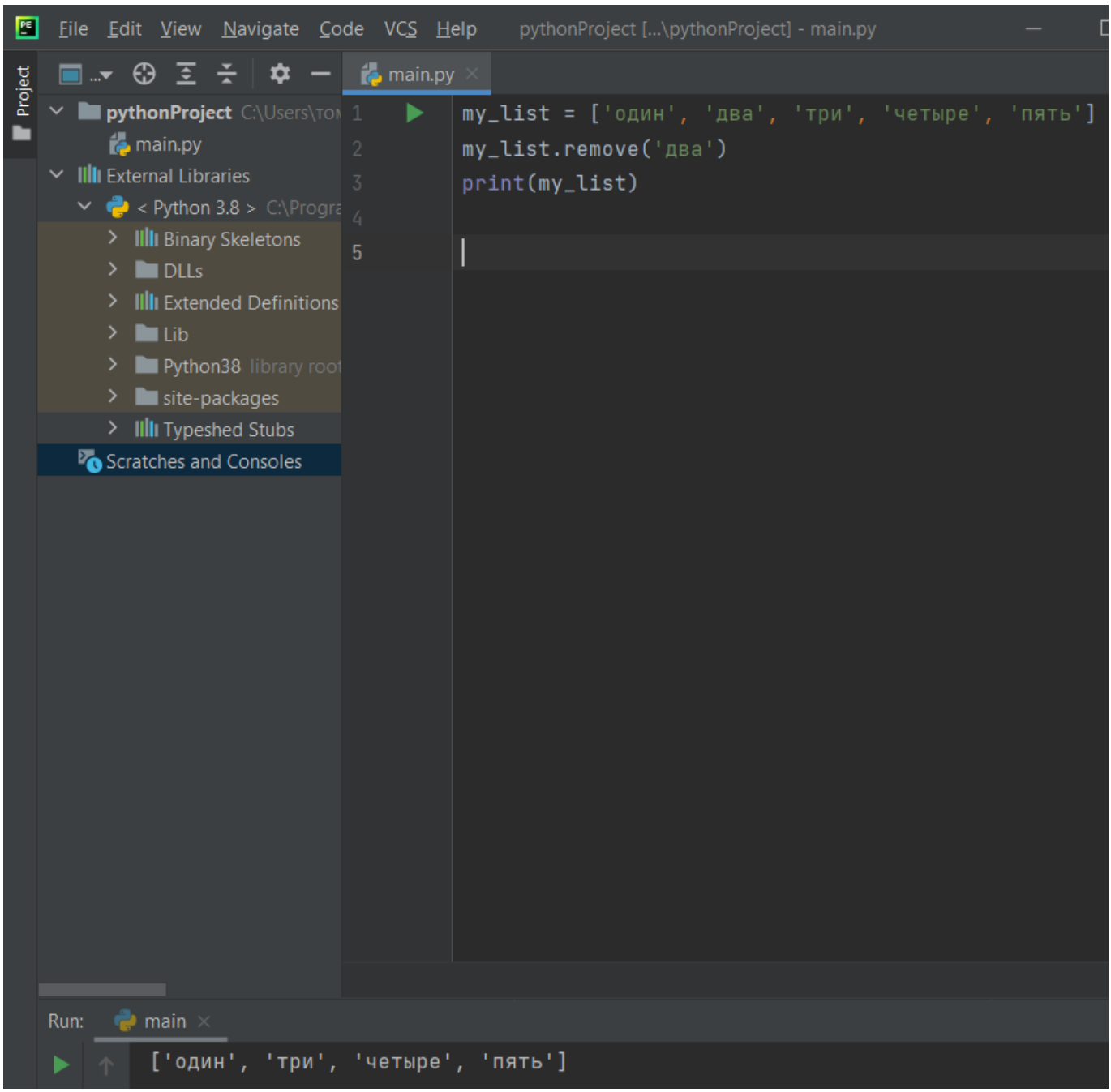

Рисунок 20 – Пример удаления элемента с помощью метода remove

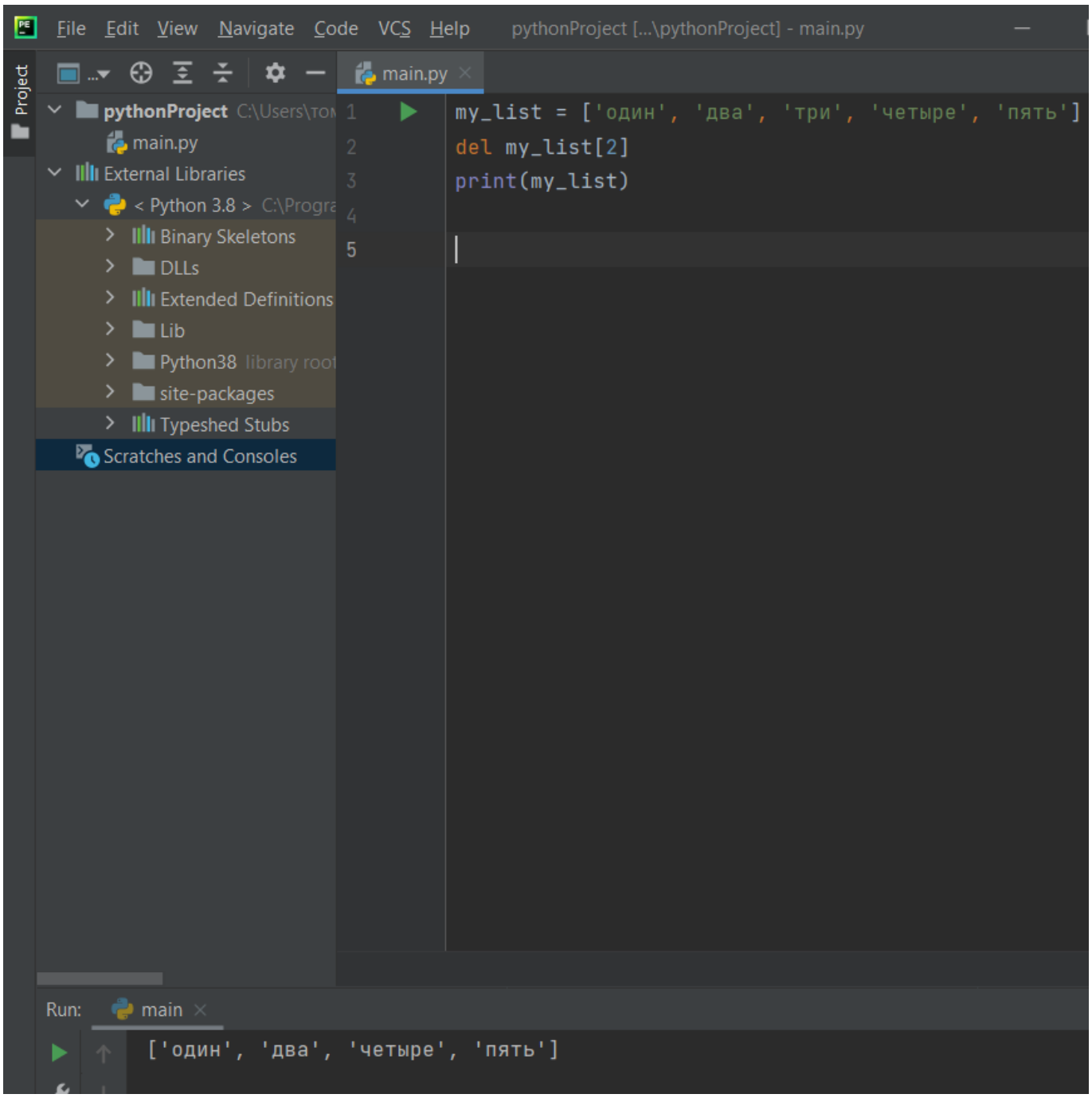

Рисунок 21 – Пример удаления элемента с помощью оператора del

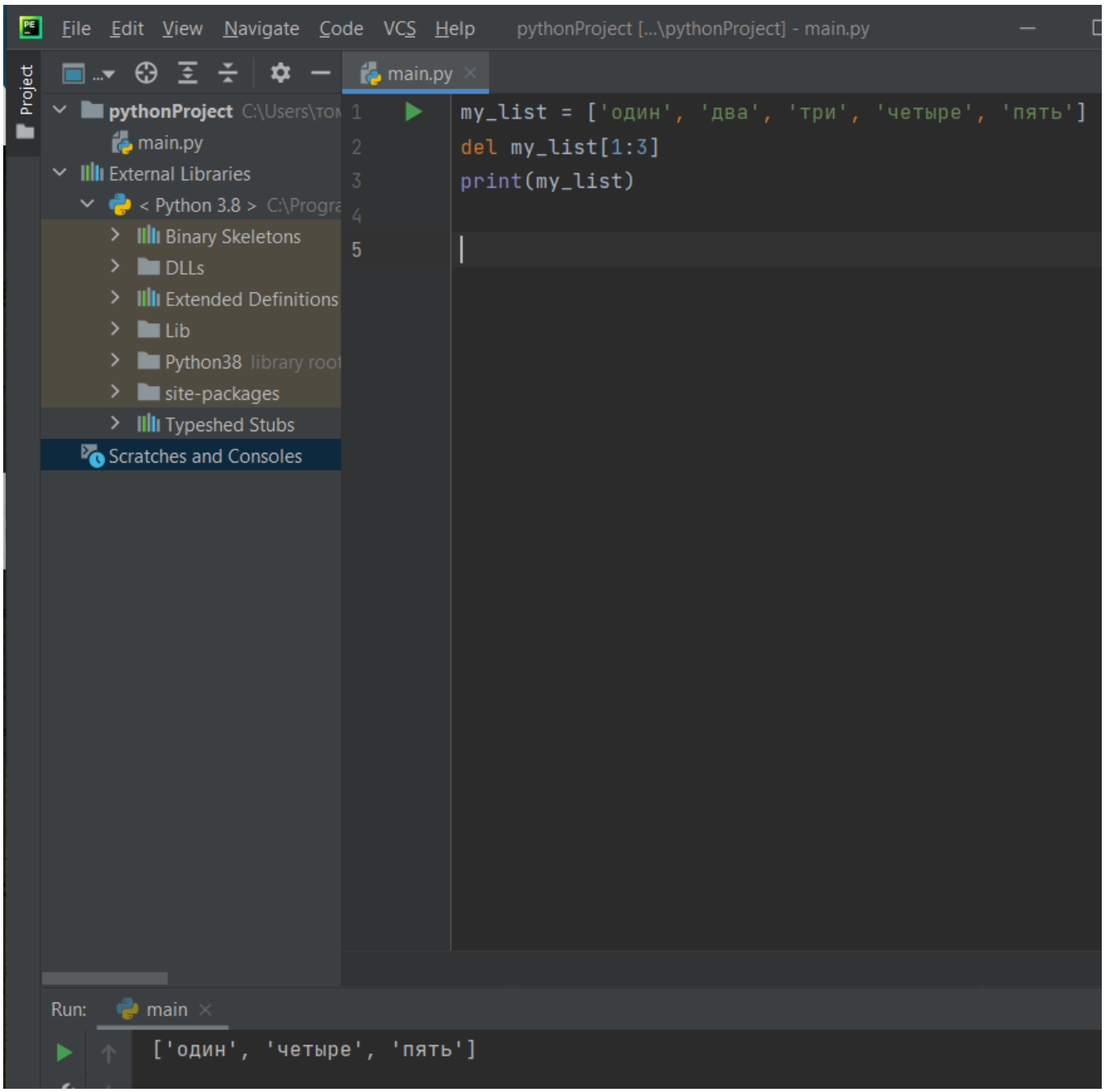

Рисунок 22 – Пример удаления нескольких элементов с помощью оператора среза

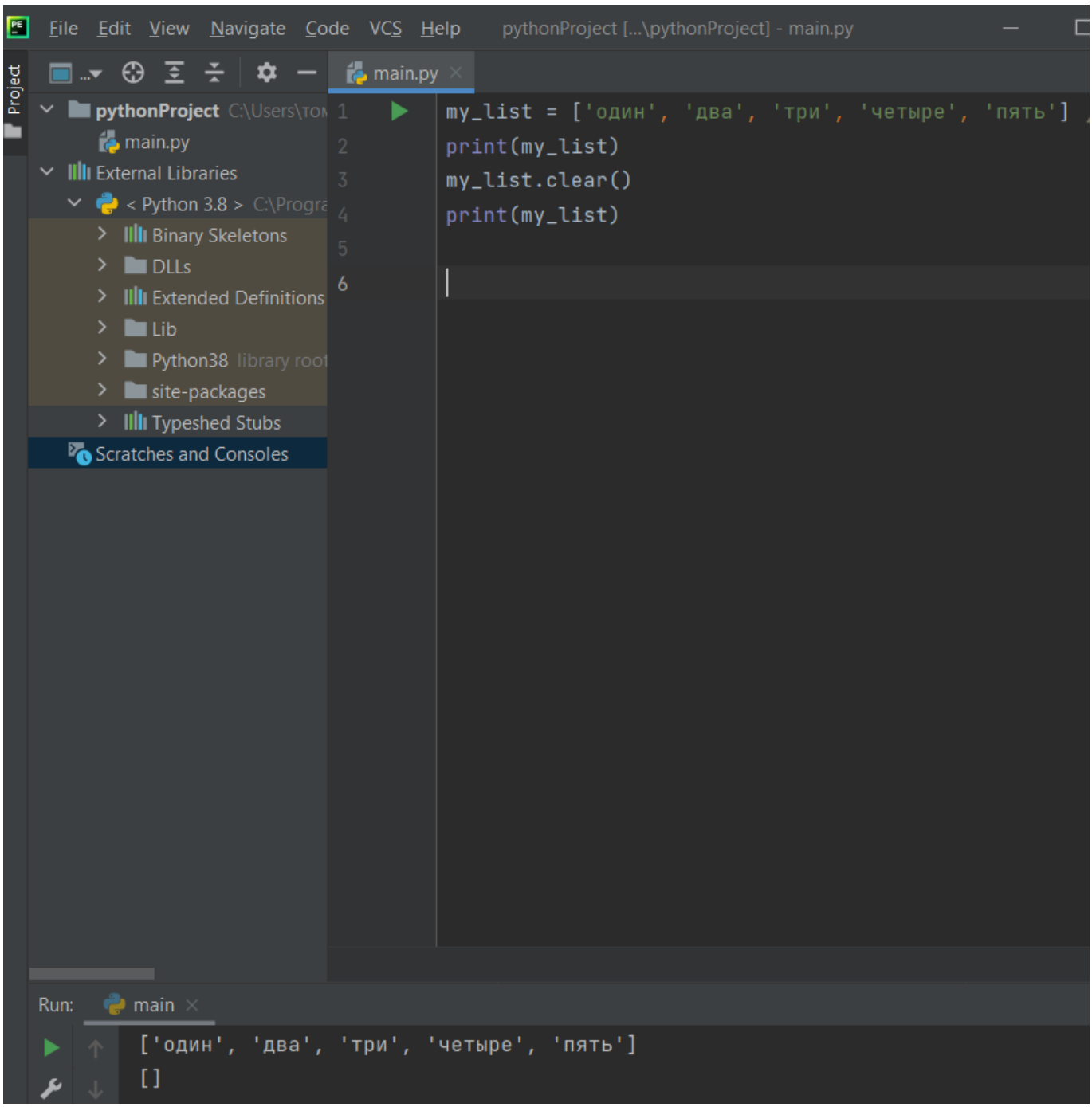

Рисунок 23 – Пример удаления всех элементов из списка

```
>>> n = int(input())
7
>>> a=[]
>>> for i in range(n):
...     a.append(i)
...
>>> print(a)
[0, 1, 2, 3, 4, 5, 6]
>>>
```

Рисунок 24 – Классический пример создания последовательности

```
>>> n = int(input())
7
>>> a = [i for i in range(n)]
>>> print(a)
[0, 1, 2, 3, 4, 5, 6]
>>>
```

Рисунок 25 – Пример работы list comprehensions

```
>>> a = [i for i in range(int(input()))]
7
>>> print(a)
[0, 1, 2, 3, 4, 5, 6]
>>>
```

Рисунок 26 – Пример работы list comprehensions, если больше не нужно использовать n

```
>>> a = [1, 2, 3, 4, 5, 6, 7]
>>> b = []
>>> for i in a:
...     b.append(i**2)
...
>>> print('a = {}\nb = {}'.format(a, b))
a = [1, 2, 3, 4, 5, 6, 7]
b = [1, 4, 9, 16, 25, 36, 49]
```

Рисунок 27 – Пример работы list comprehensions как обработчика списков

```
>>> a = [1, 2, 3, 4, 5, 6, 7]
>>> b = list(map(lambda x: x**2, a))
>>> print('a = {}\nb = {}'.format(a, b))
a = [1, 2, 3, 4, 5, 6, 7]
b = [1, 4, 9, 16, 25, 36, 49]
>>>
```

Рисунок 28 – Пример работы list comprehensions с использованием map

```
>>> a = [1, 2, 3, 4, 5, 6, 7]
>>> b = [i**2 for i in a]
>>> print('a = {}\nb = {}'.format(a, b))
a = [1, 2, 3, 4, 5, 6, 7]
b = [1, 4, 9, 16, 25, 36, 49]
```

Рисунок 29 – Пример работы list comprehensions и решением задачи через списковое включение

```
>>> a = [1, 2, 3, 4, 5, 6, 7]
>>> b = []
>>> for i in a:
...     if i%2 == 0:
...         b.append(i)
...
>>> print('a = {}\nb = {}'.format(a, b))
a = [1, 2, 3, 4, 5, 6, 7]
b = [2, 4, 6]
```

Рисунок 30 – Пример построения на базе существующего списка нового, состоящего только из чётных чисел

```
>>> a = [1, 2, 3, 4, 5, 6, 7]
>>> b = list(filter(lambda x: x % 2 == 0, a))
>>> print('a = {}\nb = {}'.format(a, b))
a = [1, 2, 3, 4, 5, 6, 7]
b = [2, 4, 6]
```

Рисунок 31 – Пример решения задачи с использованием filter

```
>>> a = [1, 2, 3, 4, 5, 6, 7]
>>> b = [i for i in a if i % 2 == 0]
>>> print('a = {}\nb = {}'.format(a, b))
a = [1, 2, 3, 4, 5, 6, 7]
b = [2, 4, 6]
>>>
```

Рисунок 32 – Пример решения задачи через списковое включение

```
>>> a = [i for i in range(10)]
>>> a[:]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> a[0:5]
[0, 1, 2, 3, 4]
>>> a[2:7]
[2, 3, 4, 5, 6]
>>> a[::2]
[0, 2, 4, 6, 8]
>>> a[1:8:2]
[1, 3, 5, 7]
>>>
```

Рисунок 33 – Пример работы слайсов и срезов

```
>>> s = slice(0, 5, 1)
>>> a[s]
[0, 1, 2, 3, 4]
>>> s = slice(1, 8, 2)
>>> a[s]
[1, 3, 5, 7]
>>>
```

Рисунок 34 – Пример работы слайса

```
>>> my_list = [5, 3, 2, 4, 1]
>>> print(len(my_list))
5
>>> print(min(my_list))
1
>>> print(max(my_list))
5
>>> print(sum(my_list))
15
>>>
```

Рисунок 35 – Пример работы функций агрегации

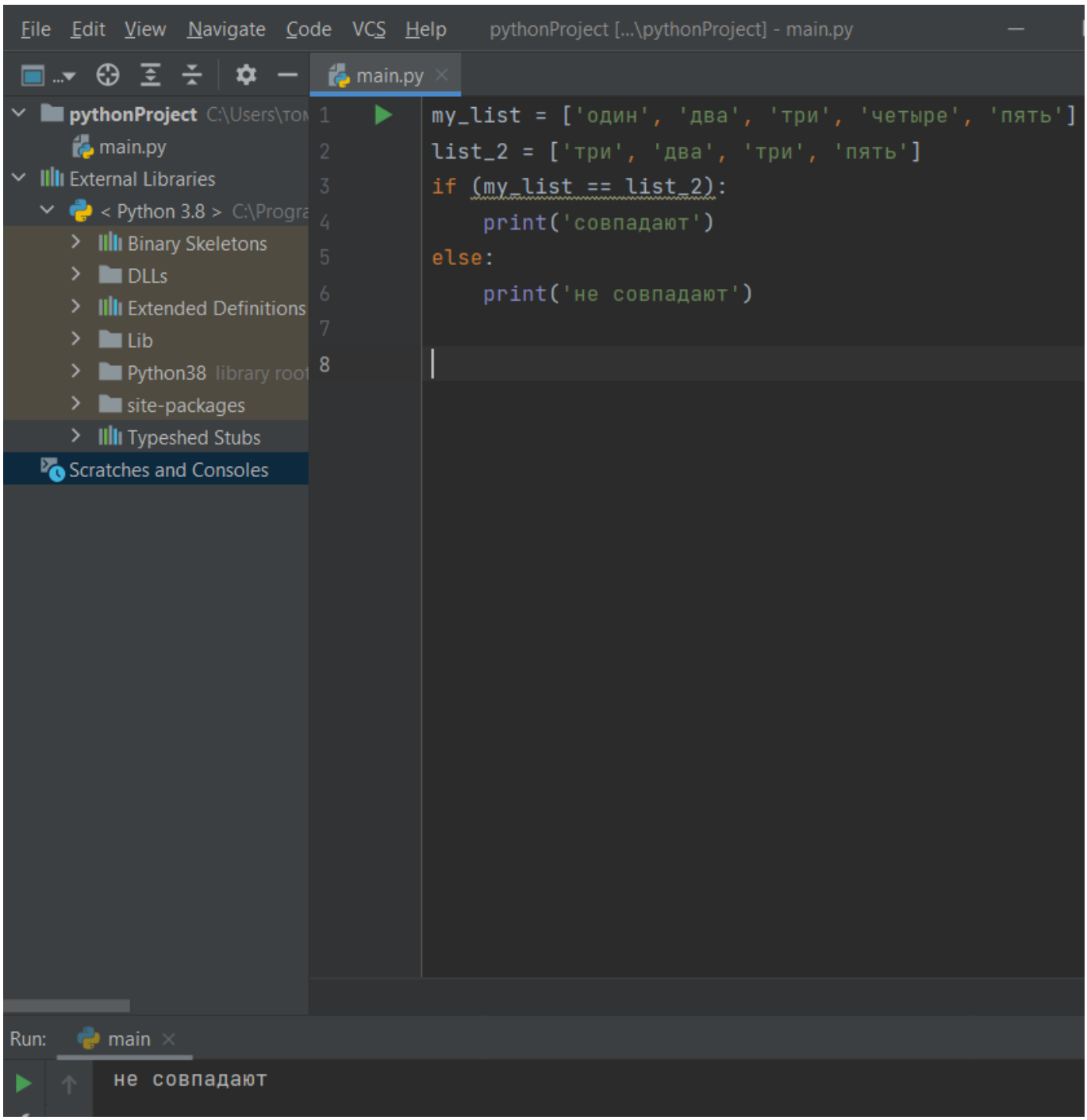

Рисунок 36 – Пример сравнения списков

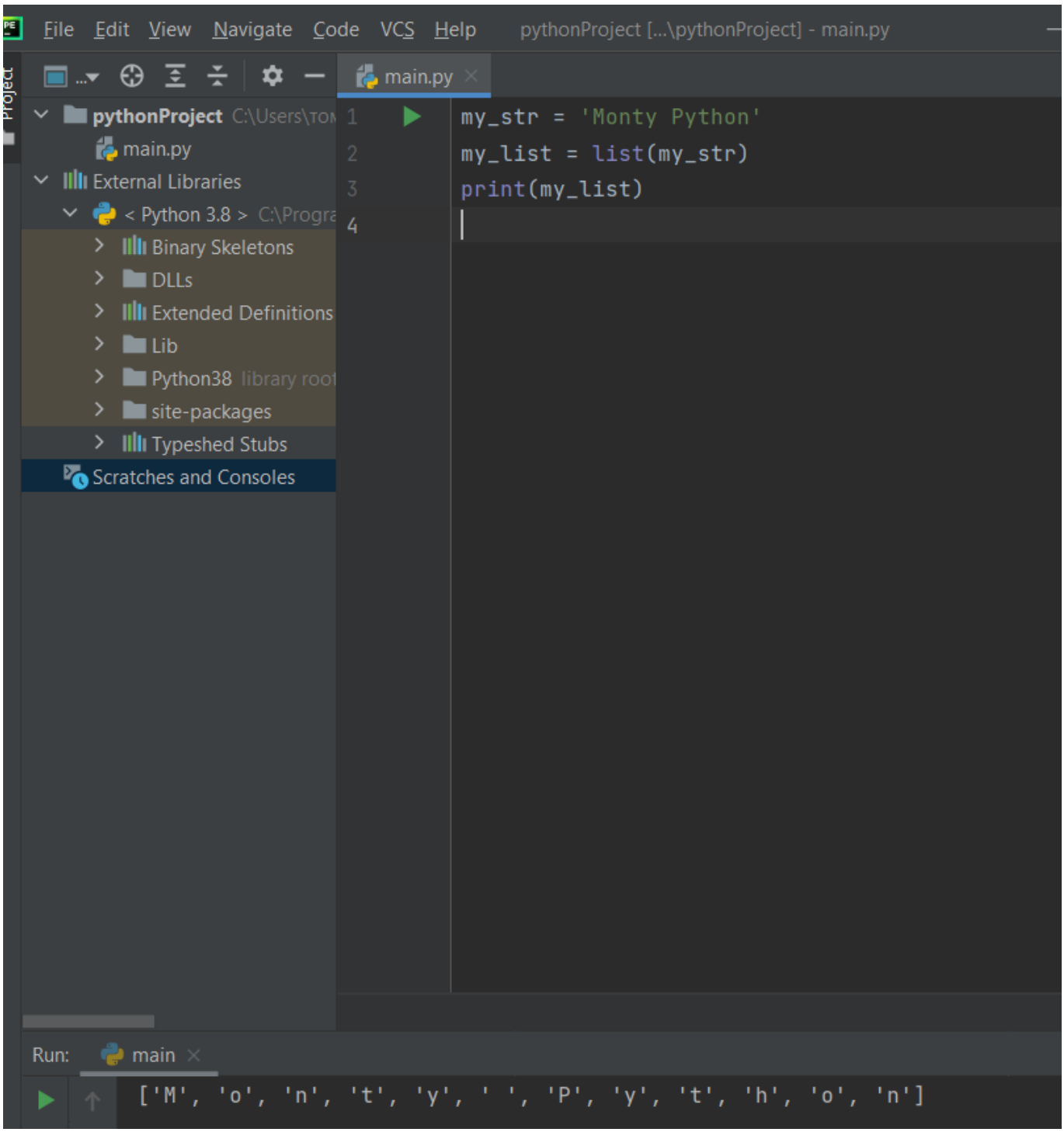

Рисунок 37 – Пример конвертации сроки в набор символов с использования функции list

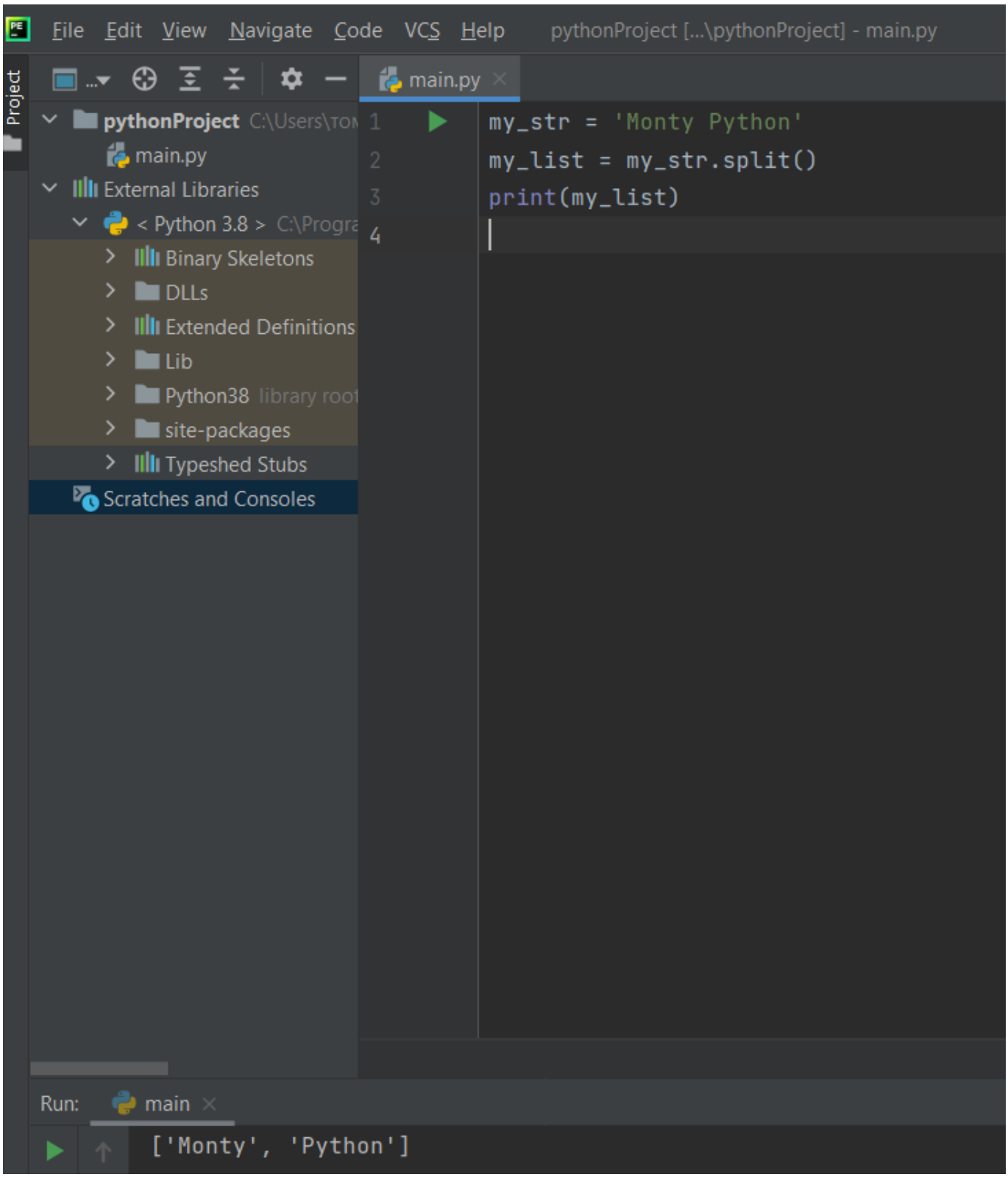

Рисунок 38 – Пример использования метода split для разбиения строки на слова

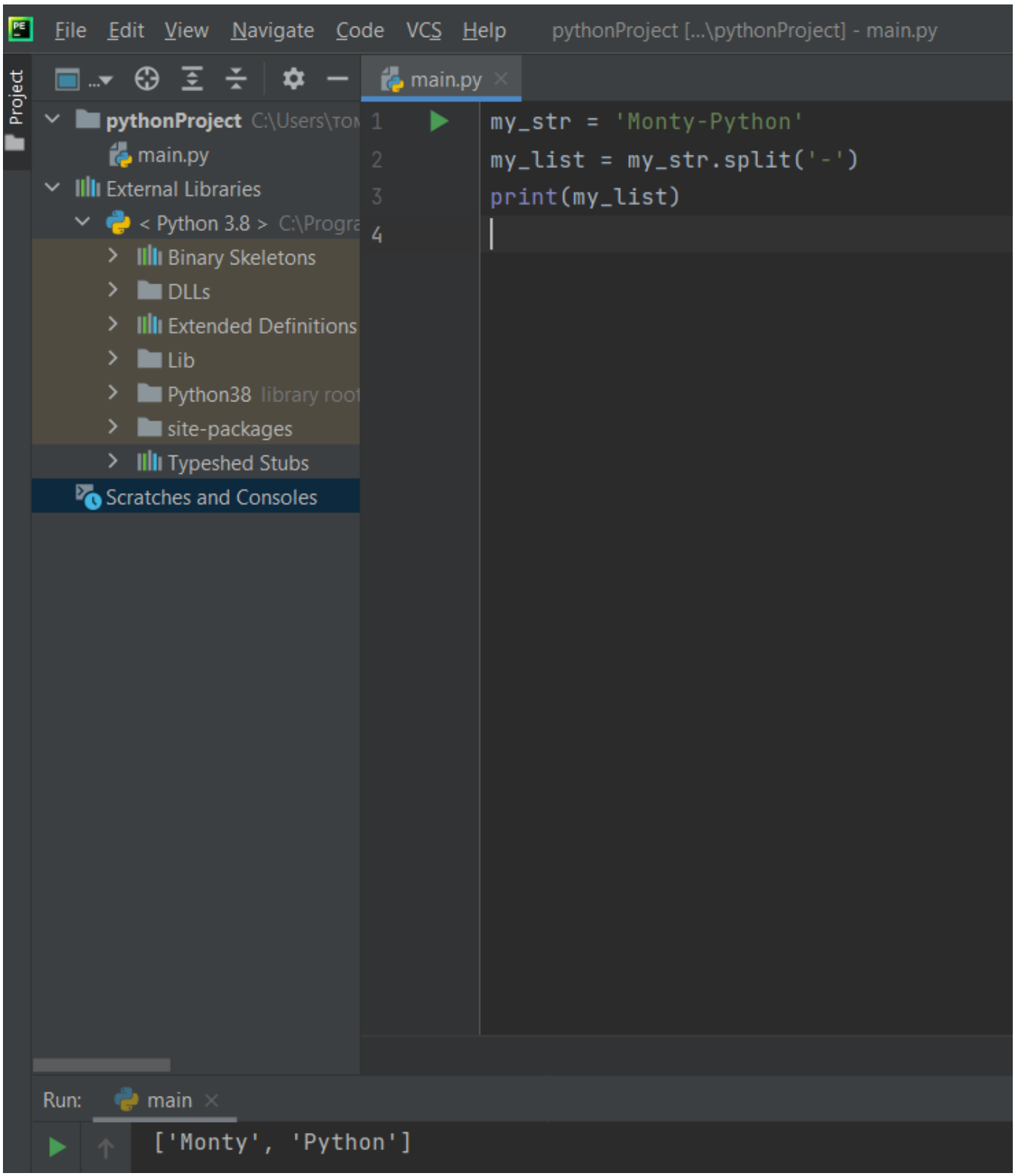

Рисунок 39 – Пример использования метода split для разбиения строки на слова, где символом разбиения служит знак «-»

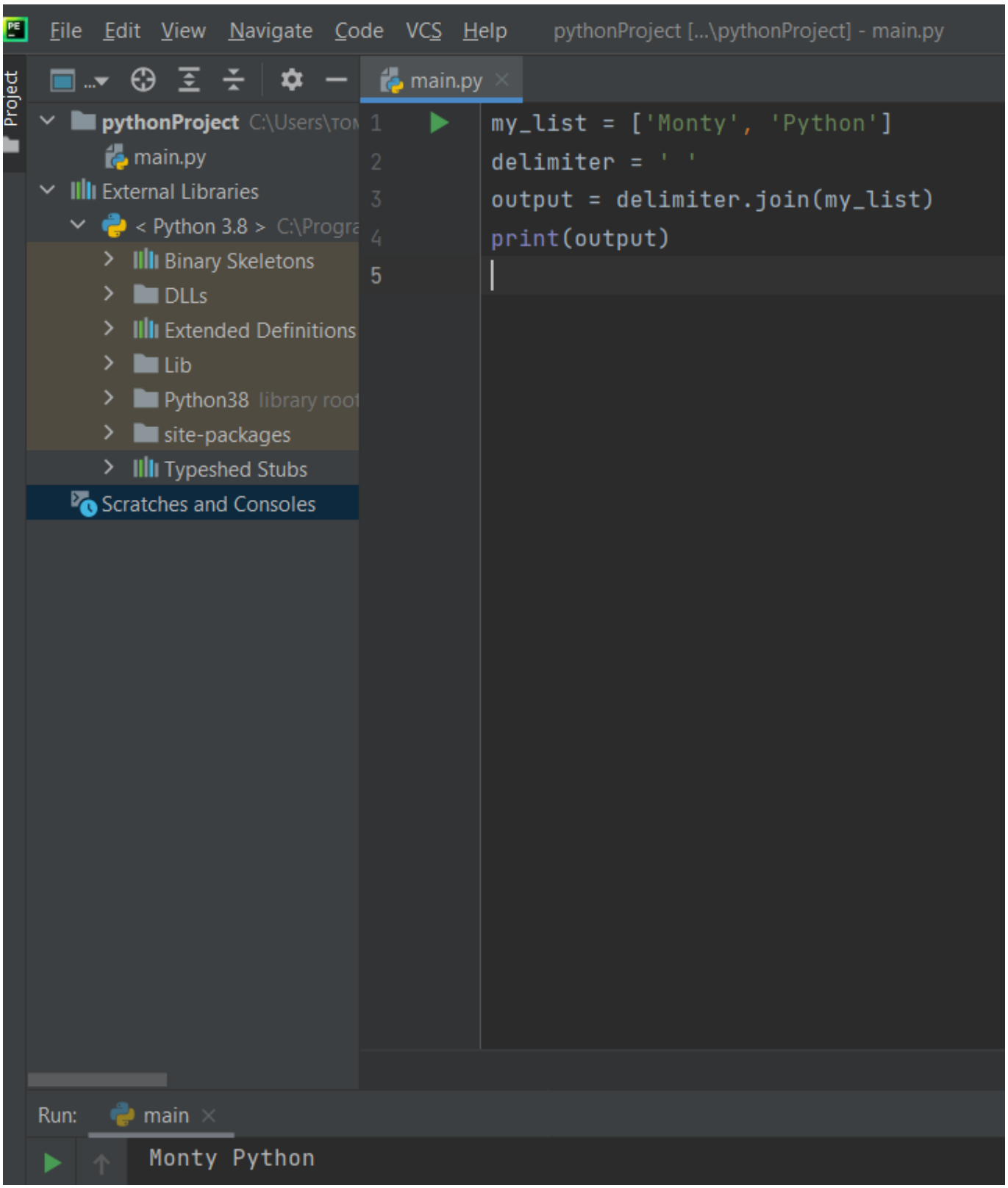

Рисунок 40 – Пример объединения элементов списка в строку

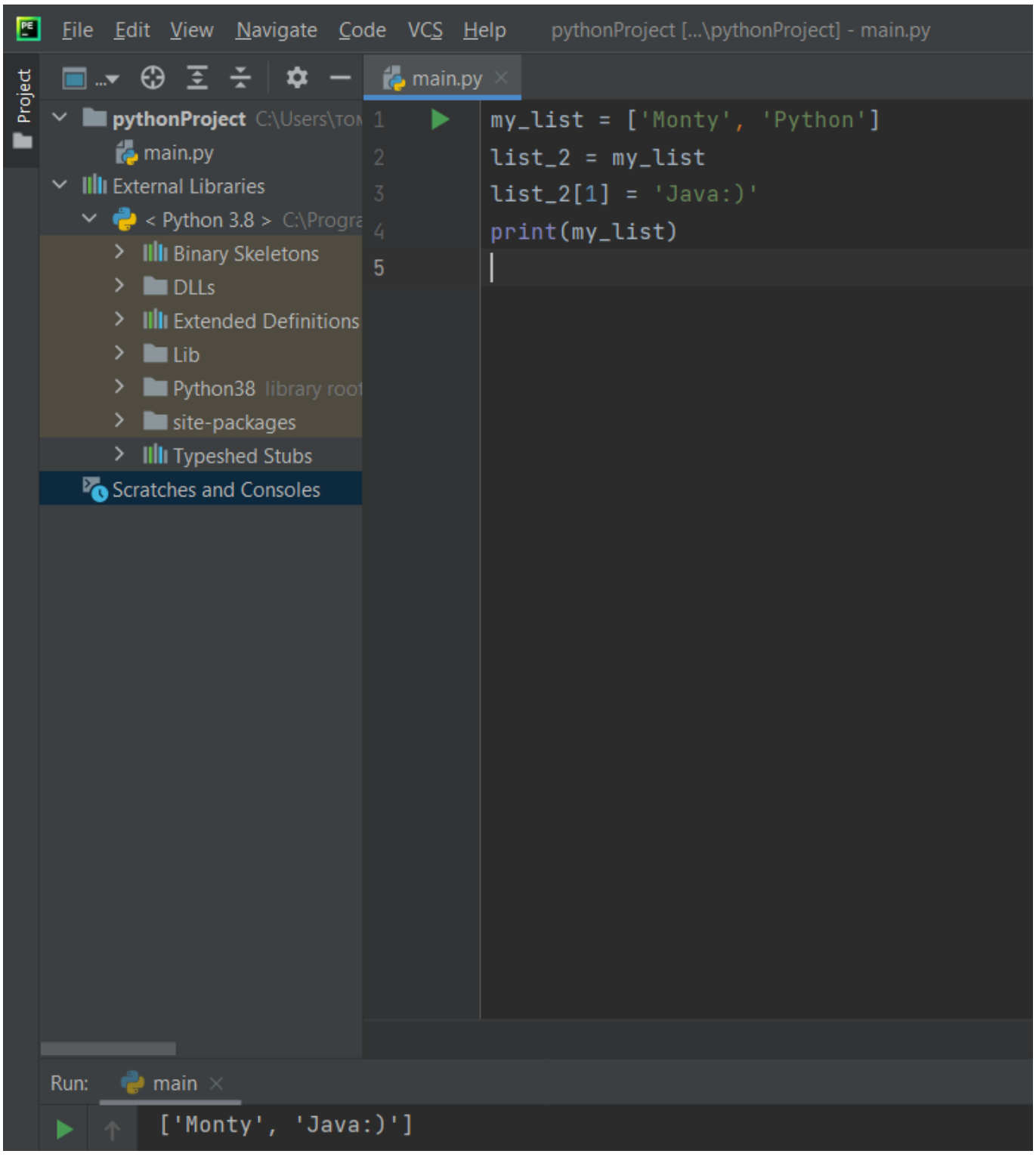

Рисунок 41 – Пример объединения элементов списка в строку

```
>>> a = [1, 2, 3, 4, 5]
>>> b = a.copy()
>>> b
[1, 2, 3, 4, 5]
>>> a == b
True
>>> a is b
False
>>> a is not b
True
>>>
```

Рисунок 42 – Пример создания копии списка с использованием метода copy

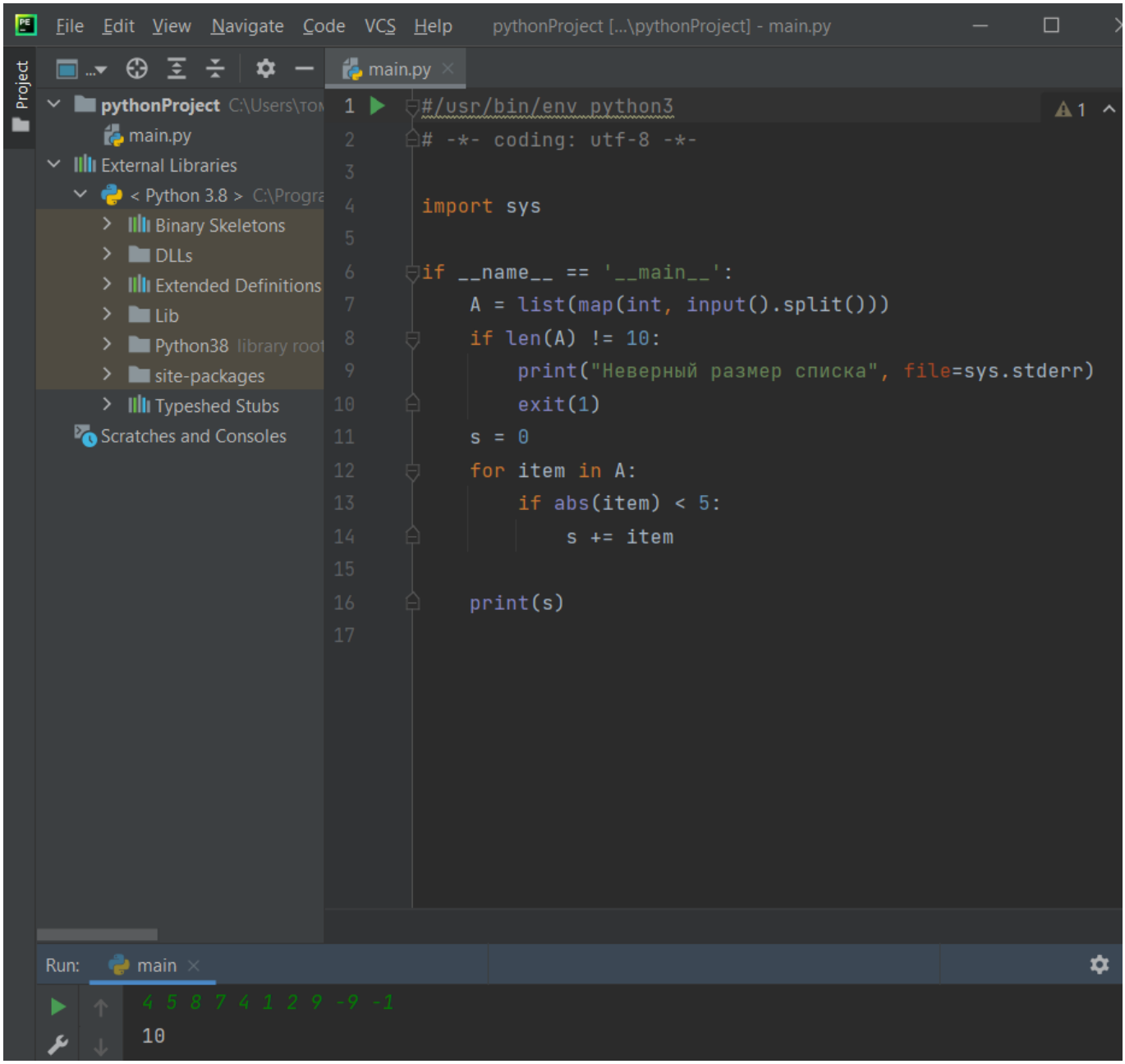

Рисунок 43 – Пример решения задачи вывода и суммы элементов списка

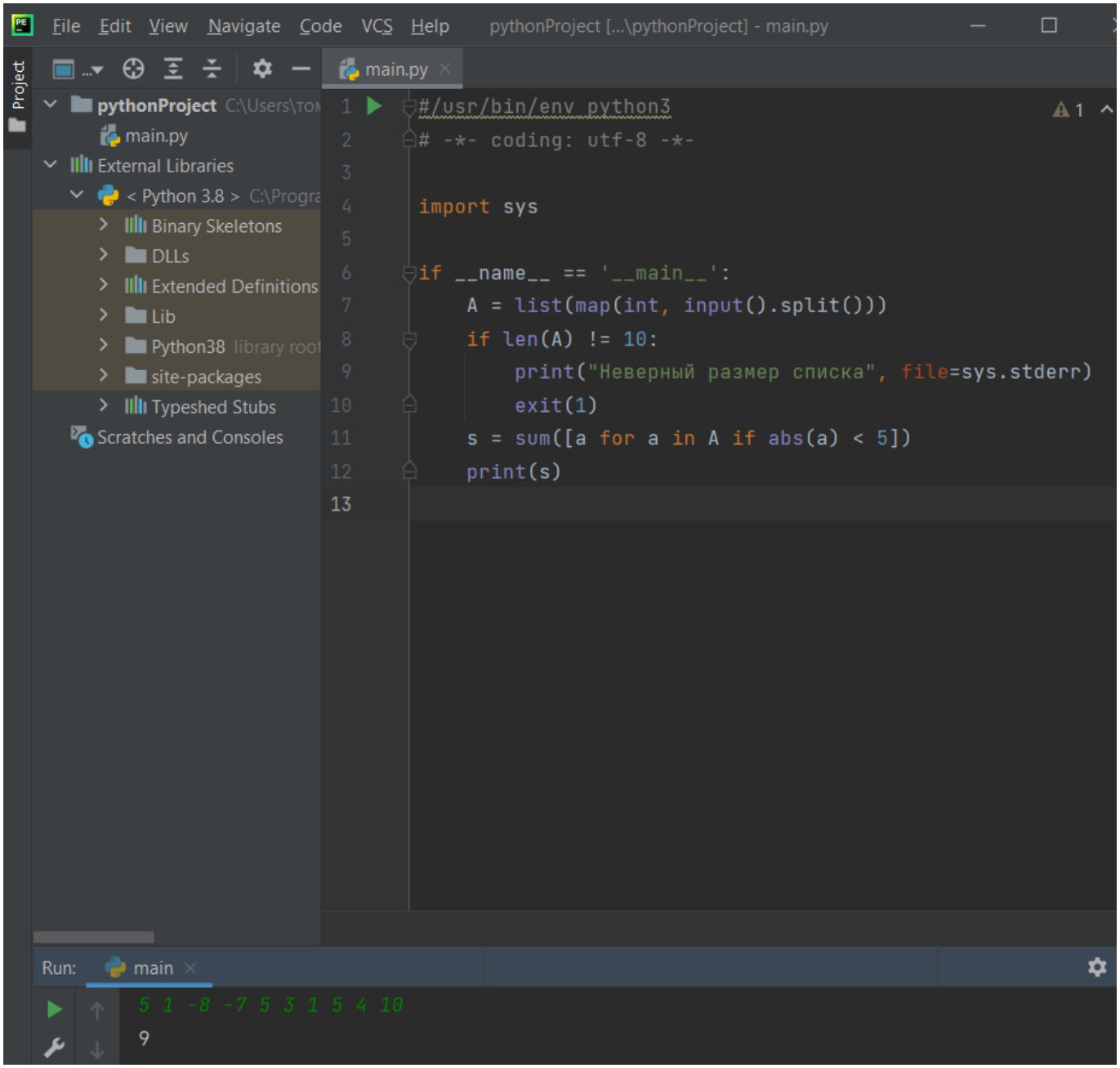

Рисунок 44 – Пример решения задачи вывода и суммы элементов списка, с помощью списковых включений

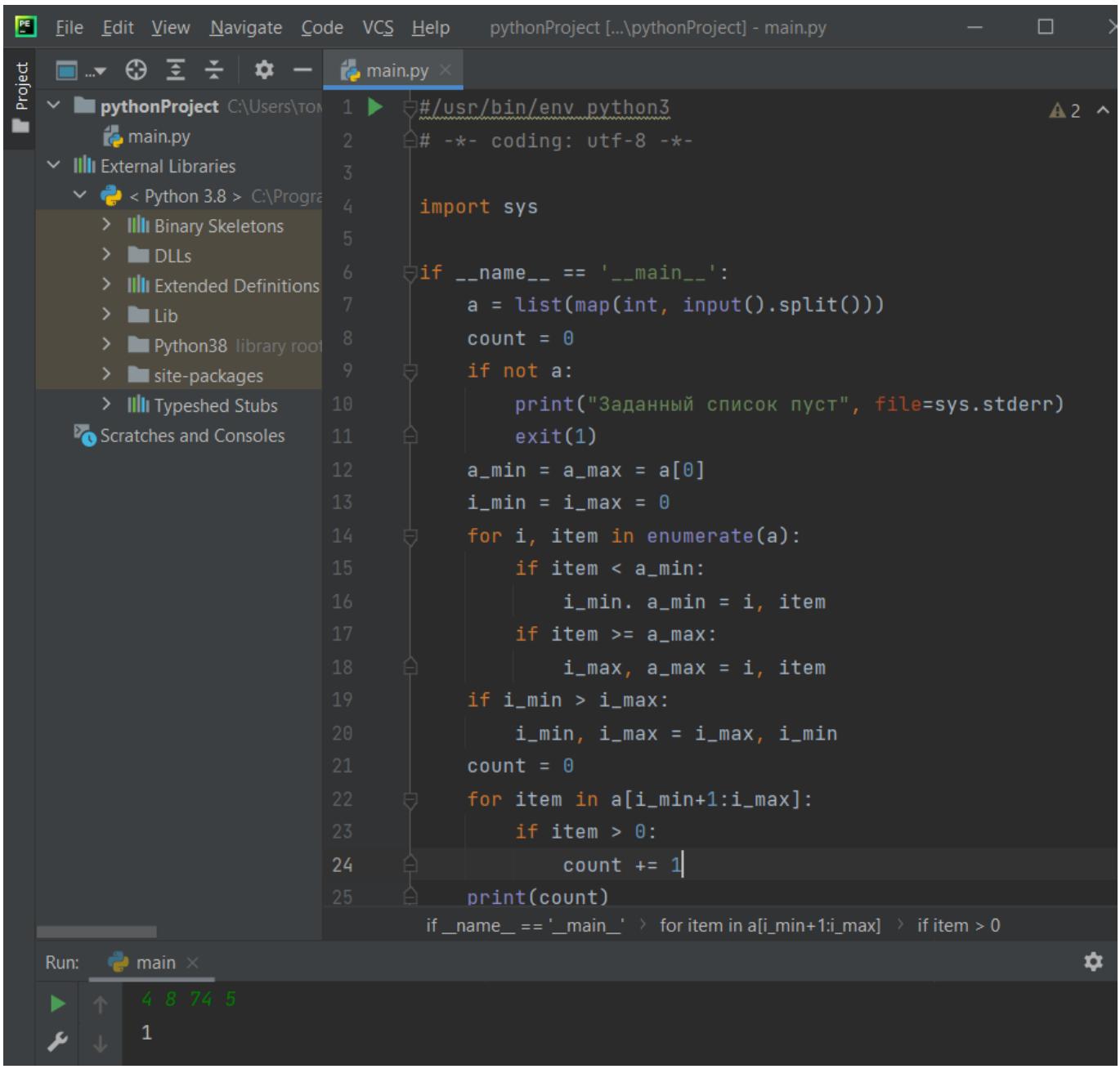

Рисунок 45 – Пример нахождения количества элементов между максимальных и минимальных элементов списка

2. Вопросы для защиты работы

1. Что такое списки в языке Python?

Список (list) – это структура данных для хранения объектов различных типов.

2. Как осуществляется создание списка в Python?

Для создания списка нужно заключить элементы в квадратные скобки.

3. Как организовано хранение списков в оперативной памяти?

Список является изменяемым типом данных. При его создании в памяти резервируется область, которую можно условно назвать некоторым “контейнером”, в котором хранятся ссылки на другие элементы данных в памяти. В отличии от таких типов данных как число или строка, содержимое “контейнера” списка можно менять.

4. Каким образом можно перебрать все элементы списка?

Элементы можно перебрать с помощью цикла for.

5. Какие существуют арифметические операции со списками?

Списки можно складывать и умножать.

6. Как проверить есть ли элемент в списке?

Проверить есть ли элемент в списке можно с помощью цикла if.

7. Как определить число вхождений заданного элемента в списке?

Чтобы определить число вхождений заданного элемента в списке, нужно воспользоваться методом count().

8. Как осуществляется добавление (вставка) элемента в список?

Метод insert можно использовать, чтобы вставить элемент в список.

Метод append можно использовать для добавления элемента в список.

9. Как выполнить сортировку списка?

Для сортировки списка нужно использовать метод sort.

10. Как удалить один или несколько элементов из списка?

Удалить элемент можно, написав его индекс в методе pop.

Элемент можно удалить с помощью метода remove.

Оператор del можно также использовать для удаления элемента.

Можно удалить все элементы из списка с помощью метода clear.

11. Что такое списковое включение и как с его помощью осуществлять обработку списков?

List Comprehensions чаще всего на русский язык переводят как абстракция списков или списковое включение, является частью синтаксиса языка, которая предоставляет простой способ построения списков.

В языке Python есть две очень мощные функции для работы с коллекциями: map и filter. Они позволяют использовать функциональный стиль программирования, не прибегая к помощи циклов, для работы с такими типами как list, tuple, set, dict и т.п. Списковое включение позволяет обойтись без этих функций.

12. Как осуществляется доступ к элементам списков с помощью срезов?

Созданный список:

```
>>> a = [i for i in range(10)]
```

Доступ к его элементам:

```
>>> # Получить копию списка
>>> a[:]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> # Получить первые пять элементов списка
>>> a[0:5]
[0, 1, 2, 3, 4]

>>> # Получить элементы с 3-го по 7-ой
>>> a[2:7]
[2, 3, 4, 5, 6]

>>> # Взять из списка элементы с шагом 2
>>> a[::2]
[0, 2, 4, 6, 8]

>>> # Взять из списка элементы со 2-го по 8-ой с шагом 2
>>> a[1:8:2]
[1, 3, 5, 7]
```

13. Какие существуют функции агрегации для работы со списками?

Для работы со списками Python предоставляет следующие функции:

len(L) - получить число элементов в списке L .

min(L) - получить минимальный элемент списка L .

max(L) - получить максимальный элемент списка L .

sum(L) - получить сумму элементов списка L , если список L содержит только числовые значения.

14. Как создать копию списка?

1. Создать псевдоним. Создать новый список и присвоить значения имеющегося.

2. Создать копию, используя метод copy.

15. Самостоятельно изучите функцию sorted языка Python. В чем её отличие от метода sort списков?

Функция sorted возвращает новый отсортированный список, который получен из итерируемого объекта, который был передан как аргумент. Функция также поддерживает дополнительные параметры, которые позволяют управлять сортировкой.

```
In [1]: list_of_words = ['one', 'two', 'list', '', 'dict']

In [2]: sorted(list_of_words)
Out[2]: ['', 'dict', 'list', 'one', 'two']
```